

SQL 2014 - Módulo III

Fernando C Spinola
357.907.328-98

SQL 2014 - Módulo III



IMPACTA
EDITORAS

Créditos

Copyright © TechnoEdition Editora Ltda.

Todos os direitos autorais reservados. Este manual não pode ser copiado, fotocopiado, reproduzido, traduzido ou convertido em qualquer forma eletrônica, ou legível por qualquer meio, em parte ou no todo, sem a aprovação prévia, por escrito, da TechnoEdition Editora Ltda., estando o contrafator sujeito a responder por crime de Violação de Direito Autoral, conforme o art.184 do Código Penal Brasileiro, além de responder por Perdas e Danos. Todos os logotipos e marcas utilizados neste material pertencem às suas respectivas empresas.

"As marcas registradas e os nomes comerciais citados nesta obra, mesmo que não sejam assim identificados, pertencem aos seus respectivos proprietários nos termos das leis, convenções e diretrizes nacionais e internacionais."

SQL 2014 - Módulo III

Coordenação Geral

Marcia M. Rosa

Coordenação Editorial

Henrique Thomaz Bruscagin

Atualização

Daniel Paulo Tamarosi Salvador

Revisão Ortográfica e Gramatical

Cristiana Hoffmann Pavan

Marcos Cesar dos Santos Silva

Diagramação

Carla Cristina de Souza

Shelida Letícia Lopes

Edição nº1 | 1696_1_WEB

Novembro/2014



*Este material constitui uma nova obra e é uma derivação da seguinte obra original, produzida por TechnoEdition Editora Ltda., em Jan/2014: SQL - Módulo III
Autoria: Marcio Henrique Guimarães Barbosa*

Sumário

Informações sobre o treinamento	11
Capítulo 1 - Instalando o SQL Server.....	13
1.1. Conceitos de banco de dados.....	14
1.2. Versões do SQL Server	15
1.3. Pré-requisitos mínimos para instalação	16
1.4. Antes de iniciar a instalação.....	17
1.5. Iniciando a instalação.....	18
Pontos principais	43
Teste seus conhecimentos	45
Mãos à obra!.....	49
Capítulo 2 - Instância e banco de dados.....	69
2.1. Instância	70
2.2. Banco de dados.....	71
2.3. Banco de dados de sistema	72
2.4. Grupo de arquivos (FileGroup).....	74
2.4.1. Tipos de grupos de arquivos	78
2.4.2. Recomendações para a divisão de arquivos	78
2.5. Criando um banco de dados	79
2.5.1. Obtendo informações sobre o banco.....	82
2.5.2. Executando o CREATE DATABASE	84
2.6. Arquivo de log	87
2.7. Modificando um banco de dados	89
2.7.1. Exemplos	91
2.8. Página de dados e extensão	94
2.9. Configurações	102
2.10. Gerenciamento de memória	107
Pontos principais	108
Teste seus conhecimentos	109
Mãos à obra!.....	113
Capítulo 3 - Gerenciando tabelas e outros objetos.....	131
3.1. Introdução	132
3.2. Criando tabelas.....	132
3.2.1. Constraints	133
3.2.2. Tipos de dados SQL Server (DataTypes)	136
3.2.3. Tabelas regulares.....	140
3.2.4. Tabelas temporárias locais	150
3.2.5. Tabelas temporárias globais	150
3.2.6. Tabelas baseadas em consultas	151
3.2.7. Tabelas particionadas	151
3.2.8. Tabelas com compressão	155
3.2.9. Tabelas baseadas em arquivos (FileTable)	156

SQL 2014 - Módulo III

3.3.	Visões (Views)	162
3.3.1.	Criação de uma visão	165
3.3.2.	Alterando a visão	166
3.3.3.	Visões indexadas	167
3.4.	Criando sequências	168
3.5.	Criando sinônimos	170
3.6.	Criando tipos	171
	Pontos principais	173
	Teste seus conhecimentos	175
	Mãos à obra!	179

Capítulo 4 - Índices..... 185

4.1.	Introdução	186
4.2.	Índices	186
4.2.1.	Estruturas de índices do SQL Server	186
4.2.2.	Índice Clustered	188
4.2.3.	Índice NonClustered	190
4.2.4.	Índice Unique	192
4.2.5.	Índice composto	193
4.2.6.	Índices comprimidos	194
4.2.7.	Índices particionados	195
4.2.8.	Pilhas	196
4.3.	Determinando a criação de um índice	196
4.3.1.	Criando índices	198
4.3.2.	Criando índices com comando	199
4.3.3.	Criando índices graficamente	205
4.4.	Manutenção de índices	207
4.4.1.	Obtendo informações sobre os índices	207
4.4.2.	Obtendo informações sobre estatísticas	208
4.5.	O otimizador e o plano de execução	215
4.5.1.	Exemplo de saída de um plano de execução	217
4.5.2.	Saídas do plano de execução	218
4.5.3.	Operadores lógicos e físicos	220
4.5.4.	Sobrepondo o otimizador	225
4.6.	Índices Full-Text	226
4.6.1.	Full Population	227
4.6.2.	Change Tracking Based Population	228
4.6.3.	Incremental Timestamp-Based Population	229
4.6.4.	Criando um catálogo FULL-TEXT	230
4.7.	Criando um índice FULL-TEXT	231
4.7.1.	Pesquisando em colunas FULL-TEXT	236
	Pontos principais	238
	Teste seus conhecimentos	239
	Mãos à obra!	243

Sumário

Capítulo 5 - Gerenciando a recuperação de dados.....	257
5.1. Introdução	258
5.2. Planejando o Backup/Restore	258
5.3. Roles para execução de backup	259
5.4. Mídia para armazenar backups.....	259
5.5. Devices de backup	260
5.5.1. Backup set, media set, media family, initial media, continuation media	262
5.5.2. Usando múltiplos backup devices	263
5.6. Atividades que não podem ser executadas durante o processo de backup.....	264
5.7. Modo de recuperação.....	264
5.7.1. Modelo de recuperação completo (FULL)	265
5.7.2. Modelo de recuperação BULKED-LOGGED	266
5.7.3. Modelo de recuperação simples (SIMPLE).....	267
5.8. Modalidades de backups.....	268
5.8.1. Backup físico frio	269
5.8.2. Backup físico quente	270
5.9. Realizando backups	271
5.9.1. Backup completo	271
5.9.2. Backup de log	277
5.9.3. Backup parcial de arquivo e de grupo de arquivo	278
5.10. Backup utilizando ambiente gráfico	279
5.11. Restauração de um backup	281
5.11.1. Restauração completa de banco de dados	282
5.11.2. Restauração de grupo de arquivos ou de arquivos de banco de dados.....	287
5.11.3. Restauração de log de banco de dados.....	288
5.11.4. Restauração de página de banco de dados	289
5.11.5. Restauração de bancos de dados de sistema (MSDB)	289
5.12. Restauração utilizando ambiente gráfico	292
5.13. Anexando e desanexando um banco de dados	296
Pontos principais	297
Teste seus conhecimentos	299
Mãos à obra!	303
Capítulo 6 - Transferência e manipulação de dados.....	329
6.1. Introdução	330
6.2. Exportando e importando dados	330
6.3. Ferramenta de transformação e cópia de dados	345
6.3.1. Bulk Copy Program (BCP).....	345
6.3.2. Bulk Insert	348
Pontos principais	352
Teste seus conhecimentos	353
Mãos à obra!	357

SQL 2014 - Módulo III

Capítulo 7 - Segurança de dados.....	367
7.1. Login e usuário	368
7.2. Principals e securables	370
7.2.1. Server Securables – Endpoints	372
7.2.2. Server Securables – Logins	376
7.2.3. Criando Logins.....	379
7.2.4. Criando usuários.....	381
7.2.5. Logins default.....	383
7.3. Gerenciando acesso à instância.....	384
7.4. Gerenciando acesso aos bancos de dados	386
7.5. Grupos de permissões criados pelo usuário	388
7.6. Grupos de permissões criados para aplicações.....	393
7.7. Permissionamento	397
7.7.1. GRANT	398
7.7.2. DENY	399
7.7.3. REVOKE.....	400
7.8. Schema	401
7.9. Credenciais	404
Pontos principais	406
Teste seus conhecimentos	407
Mãos à obra!.....	411
Capítulo 8 - Automação de tarefas, alertas e operadores.....	427
8.1. Introdução	428
8.2. Conta do SQL Server Agent	429
8.3. Configurando o envio de e-mails através do SQL Server.....	432
8.4. Configurando tarefas (Jobs).....	446
8.5. Configurando operadores (Operators)	455
8.6. Configurando alertas (Alerts)	460
8.7. Configurando múltiplos SQL Server Agents utilizando centralização	470
8.8. Fazendo cópia de todas as tarefas.....	477
8.9. Solução de problemas (Troubleshooting).....	479
Pontos principais	482
Teste seus conhecimentos	483
Mãos à obra!.....	487
Capítulo 9 - Replicação e distribuição de dados	523
9.1. Introdução	524
9.2. Transação distribuída.....	524
9.3. Replicação	526
9.3.1. Síncrona unidirecional.....	526
9.3.2. Síncrona bidirecional.....	527
9.3.3. Assíncrona unidirecional	528
9.3.4. Assíncrona bidirecional	529
9.4. Escolhendo a estratégia para deposição de dados	531
9.5. Replicação de dados no SQL Server	532
9.5.1. Metáfora da replicação	532

Sumário

9.5.2.	Publicações e artigos	533
9.5.2.1.	Filtrando dados.....	533
9.6.	Tipos de assinaturas	534
9.7.	Agentes de replicação	535
9.8.	Tipos de publicação	536
9.8.1.	Snapshot Publication.....	537
9.8.2.	Transactional Publication	537
9.8.3.	Merge Publication	538
9.8.4.	Resolução de conflitos	539
9.9.	Cenário de replicação.....	539
9.9.1.	Cenário de replicação cliente/servidor.....	540
9.9.2.	Cenário de replicação entre servidores	540
9.10.	Restrições de replicação.....	541
	Pontos principais	542
	Teste seus conhecimentos	543
	Mãos à obra!	547

	Capítulo 10 - Gerenciando um banco de dados.....	573
10.1.	Introdução	574
10.2.	Auditoria.....	574
10.2.1.	Auditoria de objetos	575
10.2.1.1.	Gatilhos	575
10.2.1.2.	AUDIT DATABASE.....	580
10.2.2.	Auditoria de segurança	580
10.3.	Checklist de atividades de um DBA	582
10.3.1.	Atividades diárias.....	582
10.3.2.	Atividades semanais	595
10.3.3.	Atividades mensais	595
10.4.	Revisão da conectividade do ambiente	596
10.5.	Monitoração do ambiente.....	599
	Pontos principais	626
	Teste seus conhecimentos	627
	Mãos à obra!	631

	Capítulo 11 - Monitorando e ajustando a performance do SQL Server.....	637
11.1.	Introdução	638
11.2.	Considerações para uma boa performance.....	638
11.3.	Fatores que afetam o tempo de resposta.....	639
11.3.1.	O que fazer para diminuir o tempo de resposta	639
11.4.	Planejando o ajuste de performance.....	641
11.4.1.	Situação atual do sistema e objetivos a serem alcançados	641
11.5.	Ajustando a performance de uma aplicação	642
11.6.	Ferramentas de monitoramento	644
11.6.1.	Windows System Monitor	644
11.6.1.1.	Contadores mais relevantes	645
11.6.1.2.	Pontos de atenção.....	649
11.6.2.	SQL Profiler.....	650

SQL 2014 - Módulo III

11.6.2.1.	Broker.....	652
11.6.2.2.	Cursor	653
11.6.2.3.	CLR.....	653
11.6.2.4.	Database.....	654
11.6.2.5.	Deprecation	654
11.6.2.6.	Errors and Warnings.....	655
11.6.2.7.	Full Text	656
11.6.2.8.	Locks	656
11.6.2.9.	Objects	657
11.6.2.10.	OLE DB.....	658
11.6.2.11.	Performance	658
11.6.2.12.	Security Audit	660
11.6.2.13.	Stored Procedures	664
11.6.2.14.	Transactions	665
11.6.2.15.	TSQL.....	666
11.6.3.	Transact SQL.....	667
11.6.3.1.	Procedimentos (Stored Procedures).....	667
11.6.3.2.	DBCC	668
	Pontos principais	669
	Teste seus conhecimentos	671
	Mãos à obra!	675
 Capítulo 12 - Alta disponibilidade.....		 703
12.1.	Introdução	704
12.2.	Log Shipping	705
12.2.1.	Configurando o Log Shipping.....	707
12.3.	Database Mirroring	722
12.4.	Always ON – SQL 2014	725
12.4.1.	Arquitetura Windows Server Failover Cluster e Always ON SQL Server 2014	726
	Pontos principais	728
	Teste seus conhecimentos	729
	Mãos à obra!	733

Informações sobre o treinamento

Para que os alunos possam obter um bom aproveitamento do curso **SQL 2014 – Módulo III**, é imprescindível que eles tenham participado do nosso curso SQL 2014 – Módulo II, ou possuam conhecimentos equivalentes.

Instalando o SQL Server

1

- ✓ Conceitos de banco de dados;
- ✓ Versões do SQL Server;
- ✓ Pré-requisitos mínimos para instalação;
- ✓ Antes de iniciar a instalação;
- ✓ Iniciando a instalação.



IMPACTA
EDITORA

1.1. Conceitos de banco de dados

Quando falamos sobre banco de dados, temos em vista apenas o conjunto de dados representados pelas tabelas, índices e demais objetos, no entanto, existem outros conceitos que devem ser tratados neste momento. Neste capítulo, abordaremos dois deles: a instância (**Instance**) e o banco de dados (**Database**).

Instância é uma instalação do SQL que atende às requisições das aplicações e dos usuários de forma independente. Possui como proposta o aumento do uso e alocação de objetos e dados em memória a fim de melhorar a organização e a performance do banco de dados. Podemos ter uma ou mais instâncias em um mesmo servidor, que podem ser classificadas como padrão (**Default**), cujo nome é **MSSQLSERVER**, ou nomeada (**Named Instance**), cujo nome deve ser atribuído no momento da criação do banco de dados. Veremos mais sobre instância em outro capítulo.

Cada instância possui recursos próprios que deverão ser compartilhados com os bancos de dados associados a ela. O banco de dados é um contêiner em disco que atende às necessidades de armazenamento físico de dados através do uso de arquivos. Ele é acessado apenas através de uma instância, exceto quando o SQL Server estiver sendo utilizado. Nesse caso, não existe a estrutura de instância. Veremos mais sobre banco de dados em outro capítulo.

1.2. Versões do SQL Server

A seguir, temos o quadro comparativo das versões SQL Server:

Recurso	Enterprise	BI.	Standard	Web	Express AS	Express Tools	Express
Número de processadores por instância operacional	Máximo do sistema operacional.	Até 4 processadores ou 16 cores.	Até 4 processadores ou 16 cores.	Até 4 processadores ou 16 cores.	Até 1 processador ou 4 cores.	Até 1 processador ou 4 cores.	Até 1 processador ou 4 cores.
Número de processadores por instância Analysis Services e Reporting Services	Máximo do sistema operacional.	Até 4 processadores ou 16 cores.	Até 4 processadores ou 16 cores.	Até 4 processadores ou 16 cores.	Até 1 processador ou 4 cores.	Até 1 processador ou 4 cores.	Até 1 processador ou 4 cores.
Máximo de memória por instância operacional	Máximo do sistema operacional.	64GB	64GB	64GB	1GB	1GB	1GB
Tamanho máximo do banco de dados	524PB	524PB	524PB	524PB	10GB	10GB	10GB

1.3. Pré-requisitos mínimos para instalação

O software de gerenciamento de banco de dados (SGBD) SQL Server 2014 que será instalado neste capítulo é da versão Enterprise. Os pré-requisitos de software para instalação são:

- Windows Server 2008 (32 ou 64 Bits) ou superior;
- Windows 7 (32 ou 64 Bits) ou superior;
- .NET Framework 3.5 e 4.0;
- Windows Installer 6.0 ou superior;
- Windows PowerShell 2.0 ou superior.

Os pré-requisitos mínimos de hardware para a instalação do SQL Server 2014 são:

- 6 GB de espaço livre em disco;
- 1 GB de memória;
- Processador x86 (1,0GHz) ou x64 (1,4GHz).

1.4. Antes de iniciar a instalação

Antes de iniciar o processo de instalação, recomenda-se a execução de alguns passos preparatórios a fim de podermos restaurar o ambiente em caso de falha do processo de instalação.

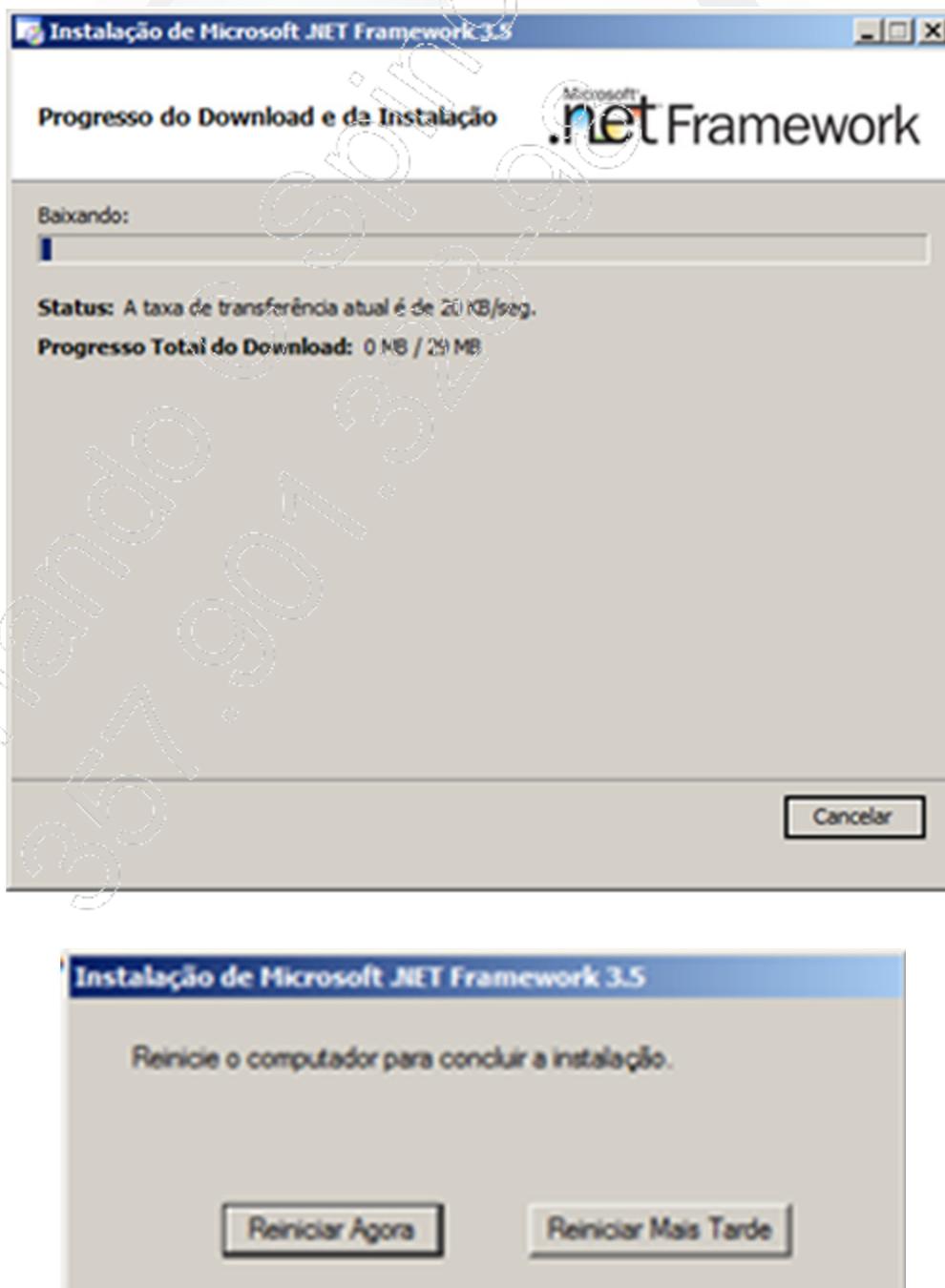
- **Passo 1 - Verificação da fragmentação dos discos:** Nesta etapa, faremos a verificação de fragmentação dos discos. Caso os discos tenham sido recentemente instalados ou apresentados ao sistema operacional, esta operação poderá ser ignorada;
- **Passo 2 - Cópia das entradas do Regedit:** Regedit é uma ferramenta do sistema operacional Windows responsável pelo registro e manutenção de informações a respeito dos softwares instalados. A cada novo software instalado o regedit sofre atualização. É recomendável, então, realizar uma cópia de cada pasta, pois, em um cenário de reinstalação, podemos restaurar uma cópia dessa ferramenta;
- **Passo 3 - Criar um ponto de restauração antes de iniciar o procedimento de instalação:** Este procedimento será muito útil para voltarmos o cenário da instalação a um momento anterior ao seu início;
- **Passo 4 - Copiar todos os binários para instalação, incluindo service packs e patches de correção:** Havendo espaço disponível em disco, recomenda-se inclusive mantê-los em disco;
- **Passo 5 - Criação de um usuário para inicialização dos serviços:** É recomendável, por medida de segurança, que seja criado um usuário (preferencialmente no Active Directory, também conhecido como AD) que tenha um baixo nível de privilégios para ser associado aos serviços do SQL Server, a fim de inicializá-los e encerrá-los;
- **Passo 6 - Definir os discos para configuração do SQL Server:** Recomenda-se ter pelo menos três discos para a instalação do SQL Server: um para o Sistema Operacional, outro para a instalação do SQL Server e outro para a instalação da instância. Esta etapa é opcional nos casos de instâncias usadas para bancos de dados cuja missão não seja crítica;

- **Passo 7 - Definir as permissões nos discos do SQL Server:** É recomendável definir essas permissões nos discos para que sejam acessados apenas pelo usuário que realiza a instalação, impedindo, assim, acessos indevidos aos bancos de dados (databases). Esta é uma medida de segurança.

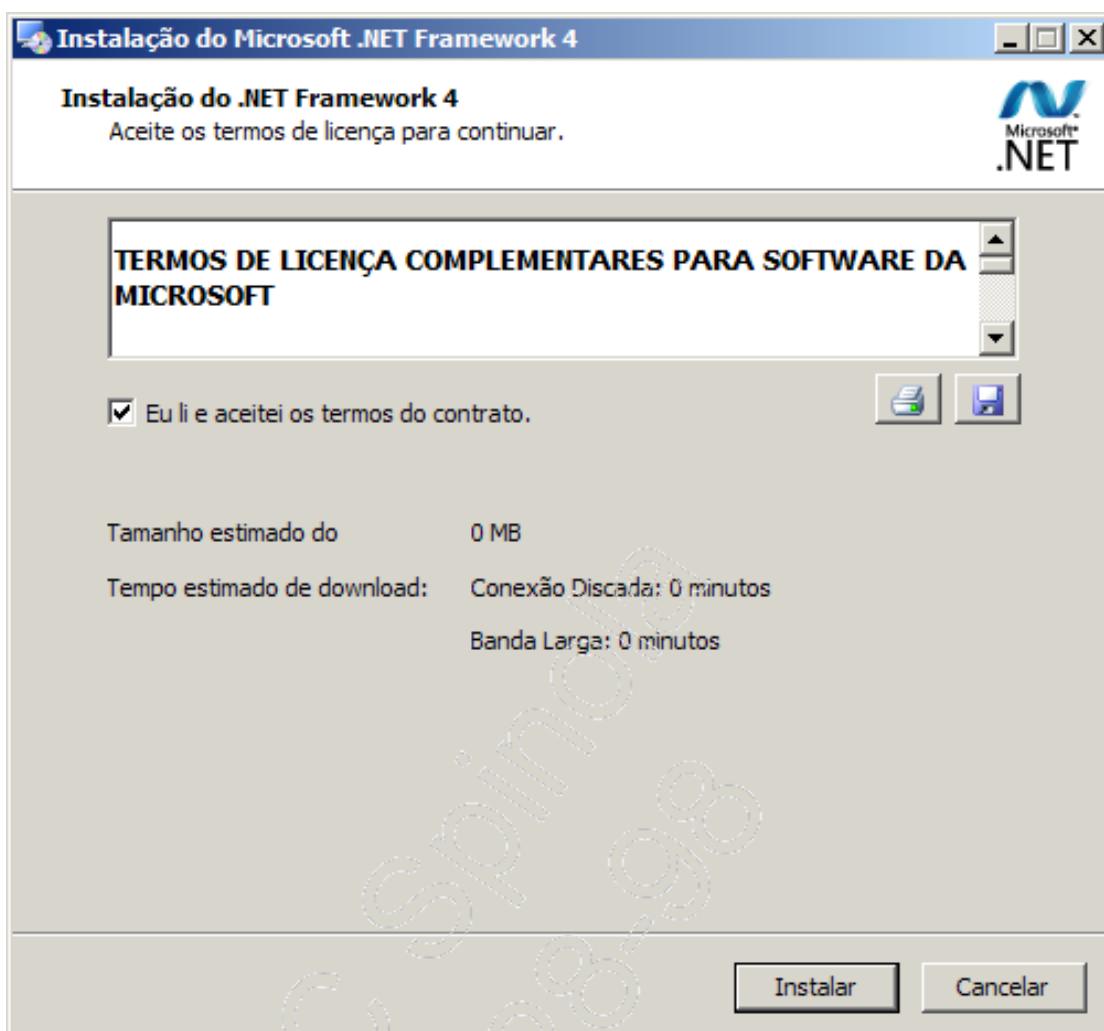
1.5. Iniciando a instalação

Realizaremos a instalação do software de gerenciamento de banco de dados SQL Server 2014, porém, vamos antes instalar o .NET 3.5, .NET 4.0 e o Powershell 3.0.

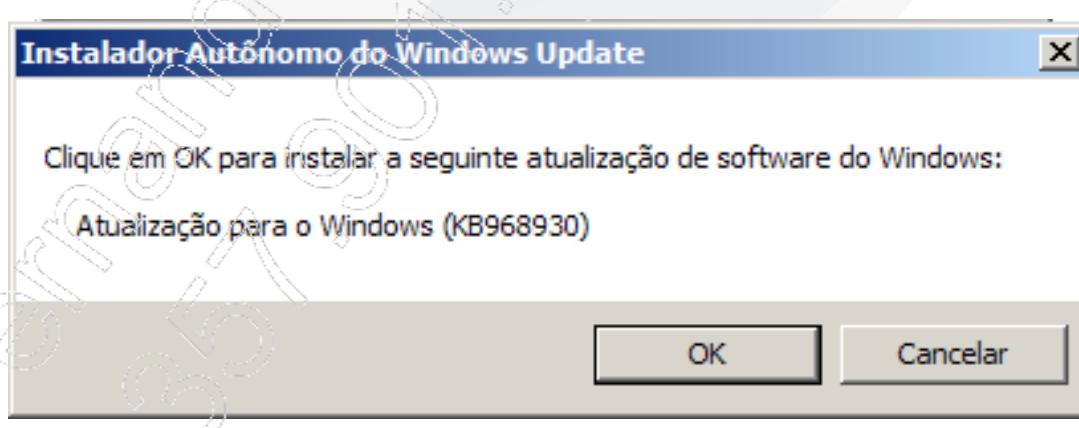
- **Instalando o .NET 3.5**



- Instalando o .NET 4.0



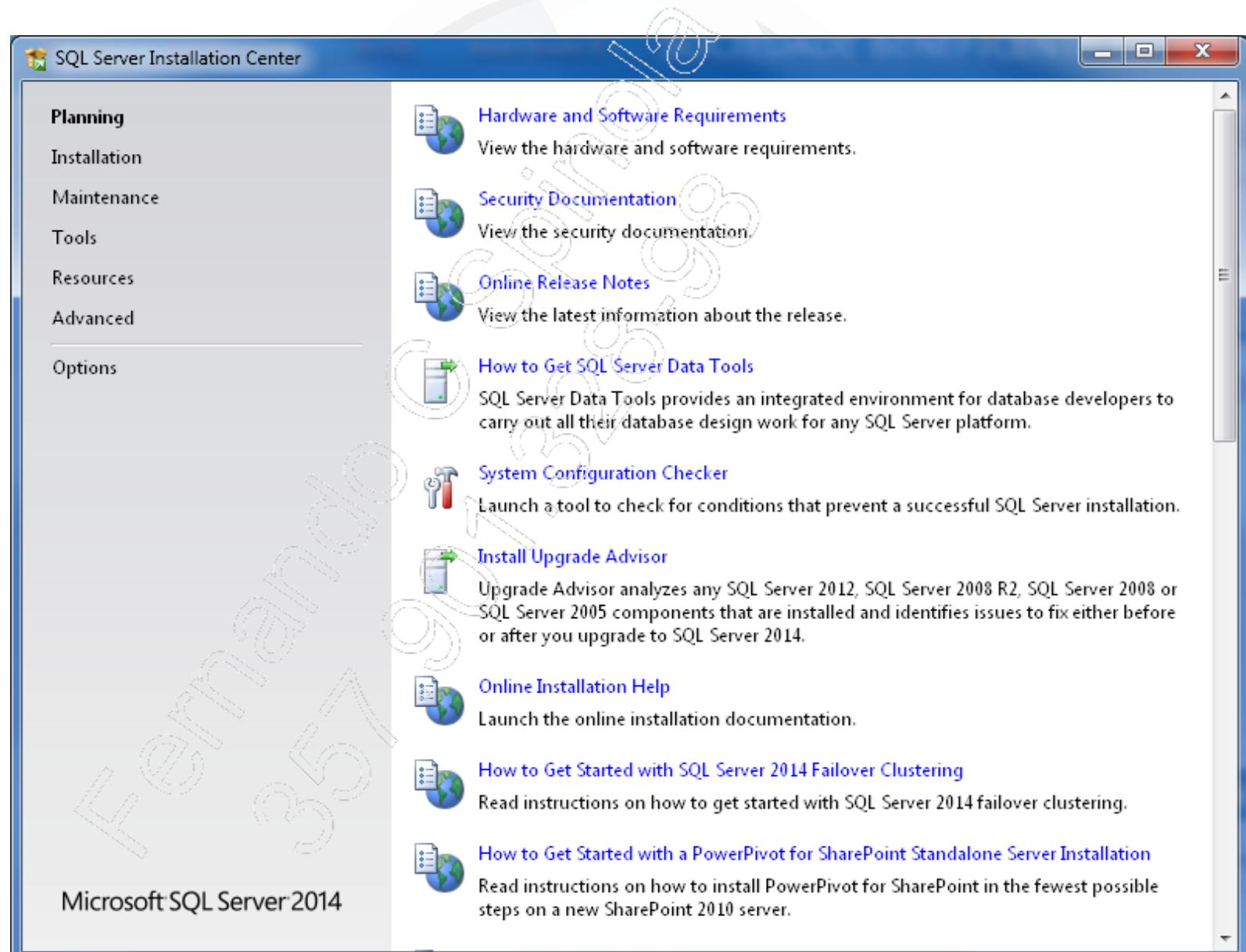
- Instalando o Powershell



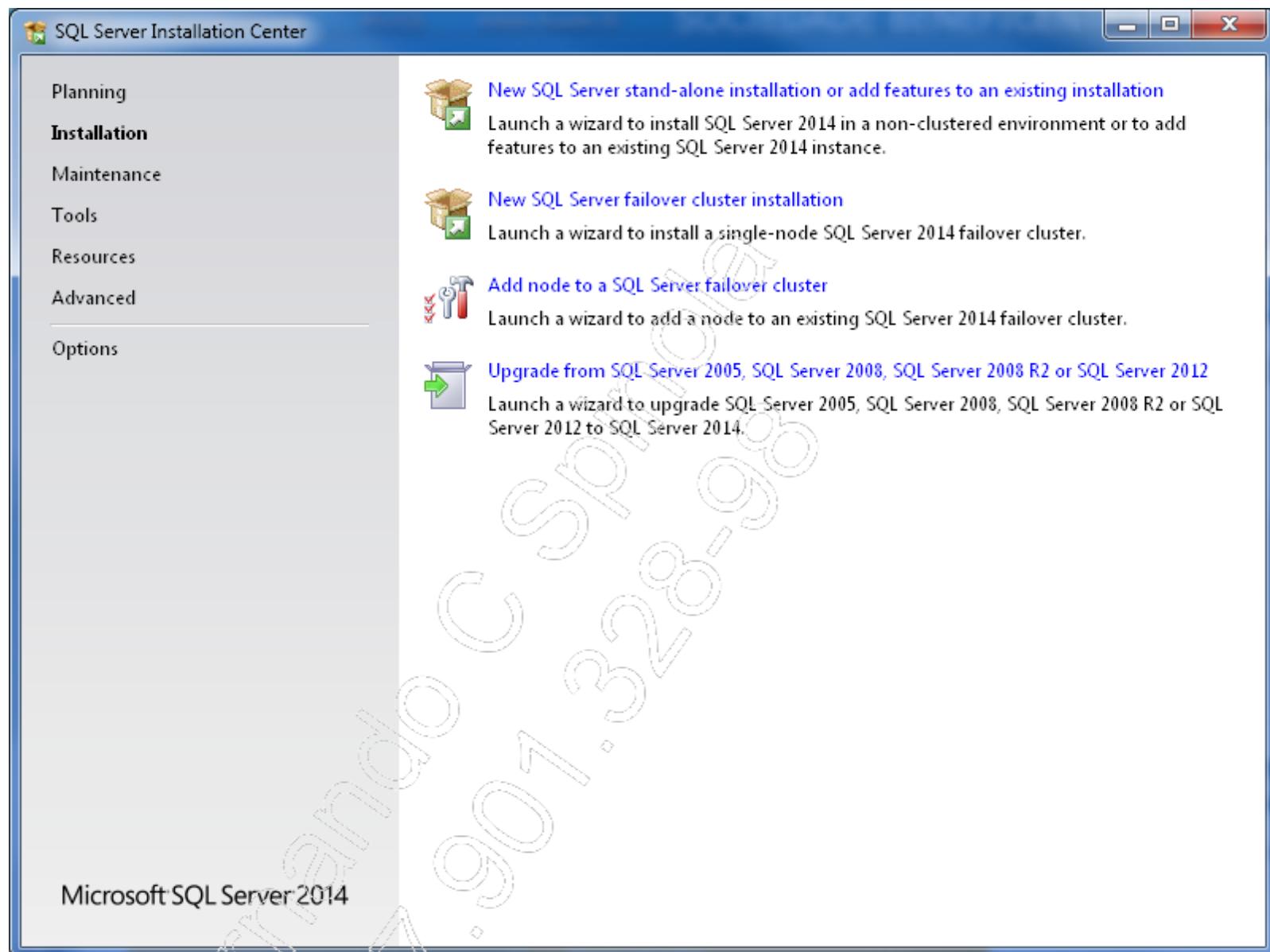
Após serem instalados os pré-requisitos para o SQL Server, vamos iniciar o procedimento de instalação. Para isso, vamos realizar a instalação da instância **Default**.

SQL 2014 - Módulo III

Descompacte o instalador e execute como administrador (RunAs).



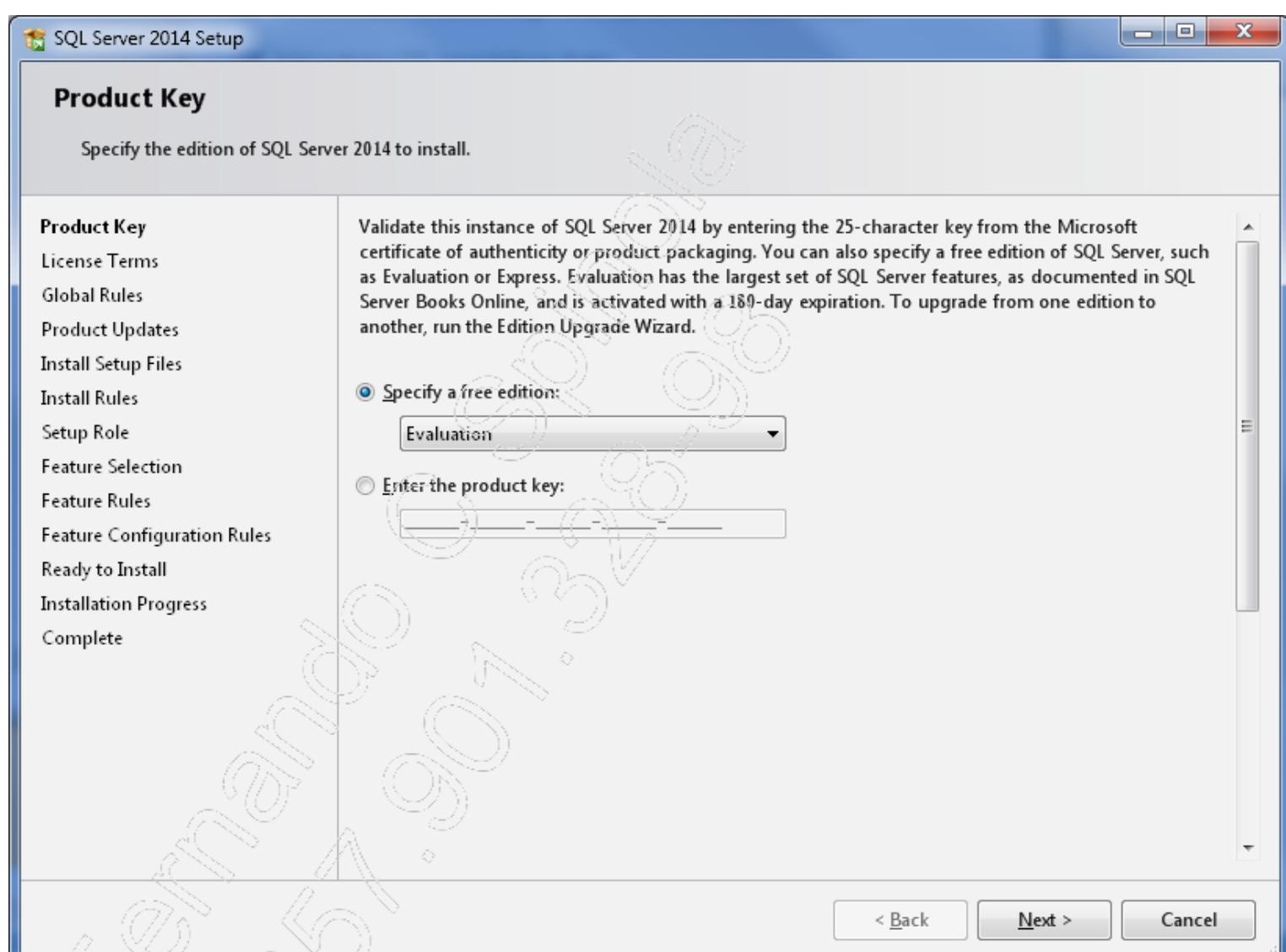
Selecione a opção **Installation**.



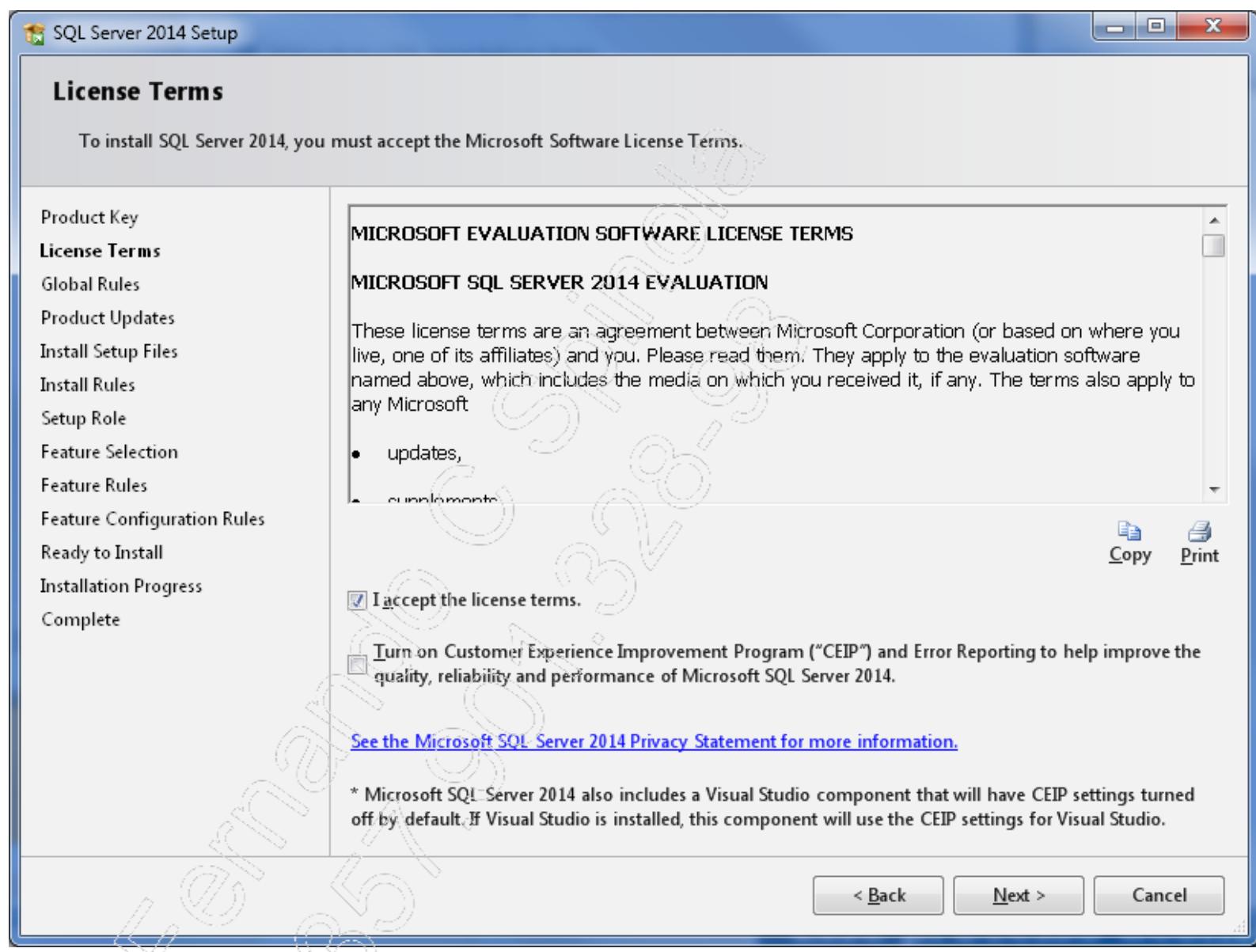
Selecione a opção **New SQL Server stand-alone installation....** Nesta etapa, nenhum processo pode falhar, pois, se isso acontecer, somente após recuperarmos a falha é que será possível restabelecer o procedimento de instalação.

SQL 2014 - Módulo III

Especifique **Evaluation** para a chave de produto e clique em **Next**.

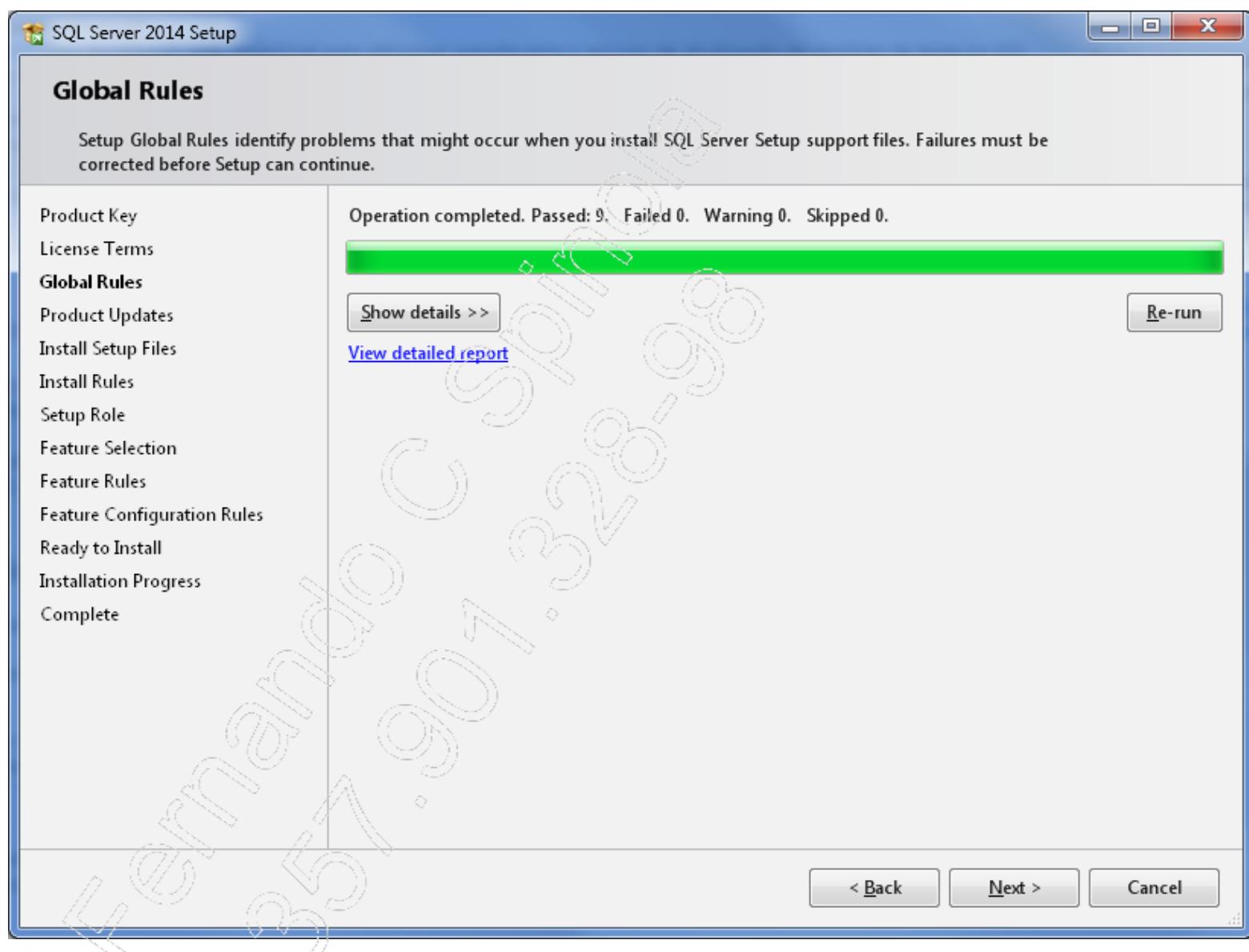


Aceite os termos da licença e clique em **Next**.

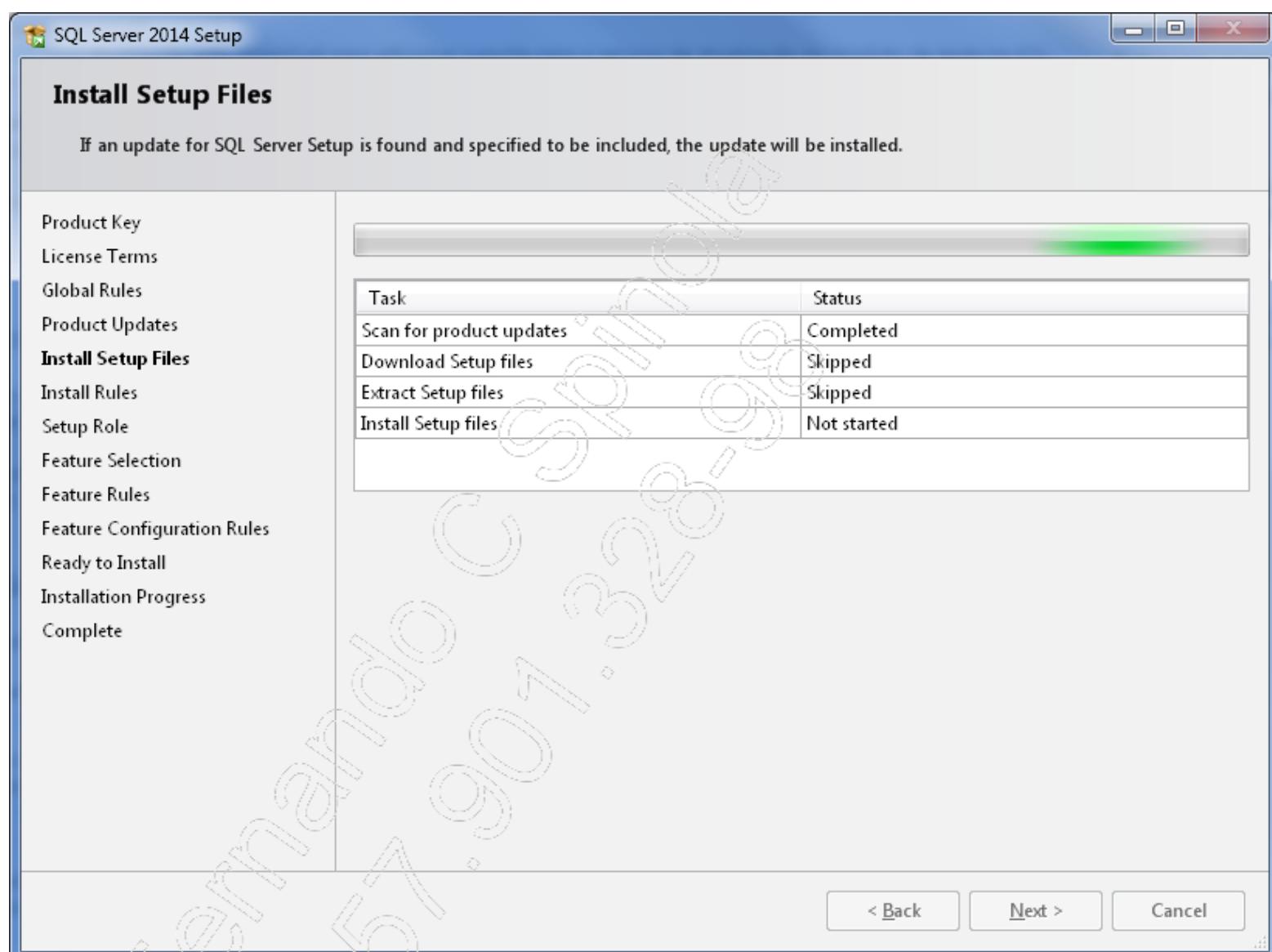


SQL 2014 - Módulo III

Clique em **Next** para a verificação do Global Setup, que analisará o processo de instalação.

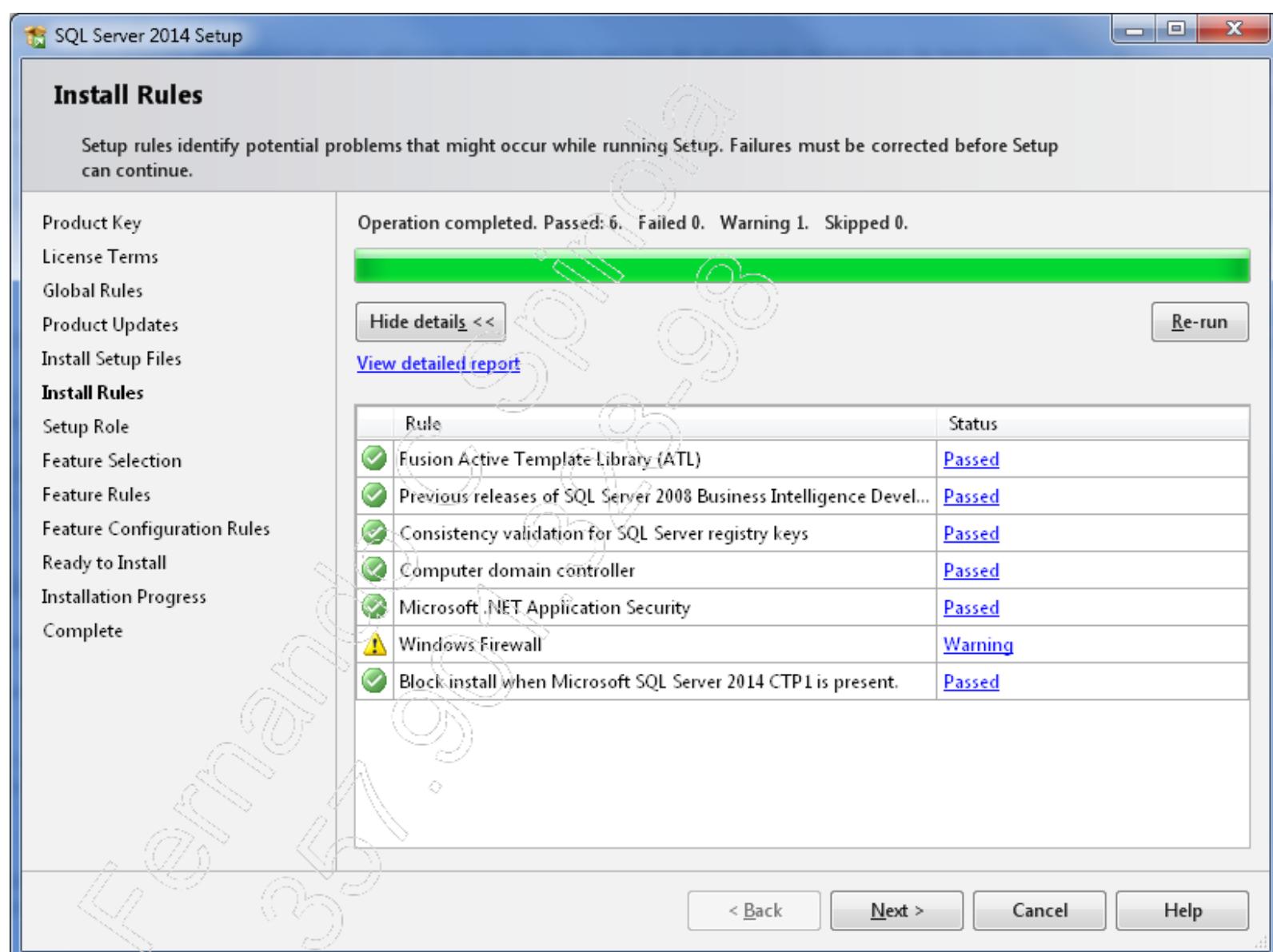


Clique em **Next** para a tela de pesquisa de atualizações.

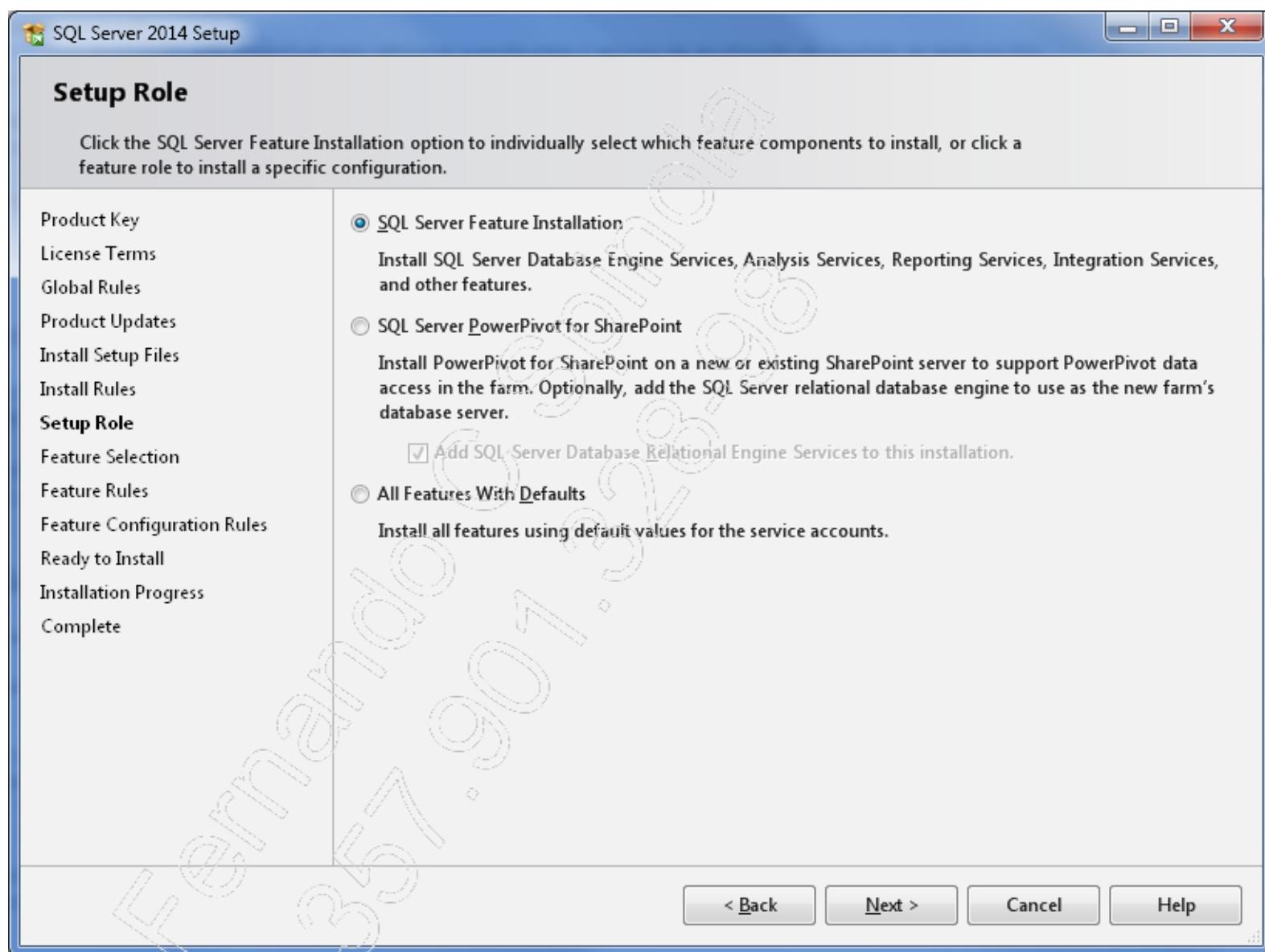


SQL 2014 - Módulo III

Verifique as regras de suporte. Normalmente a regra Windows Firewall aponta status de Warning em função da configuração de Firewall ativada junto ao Windows.

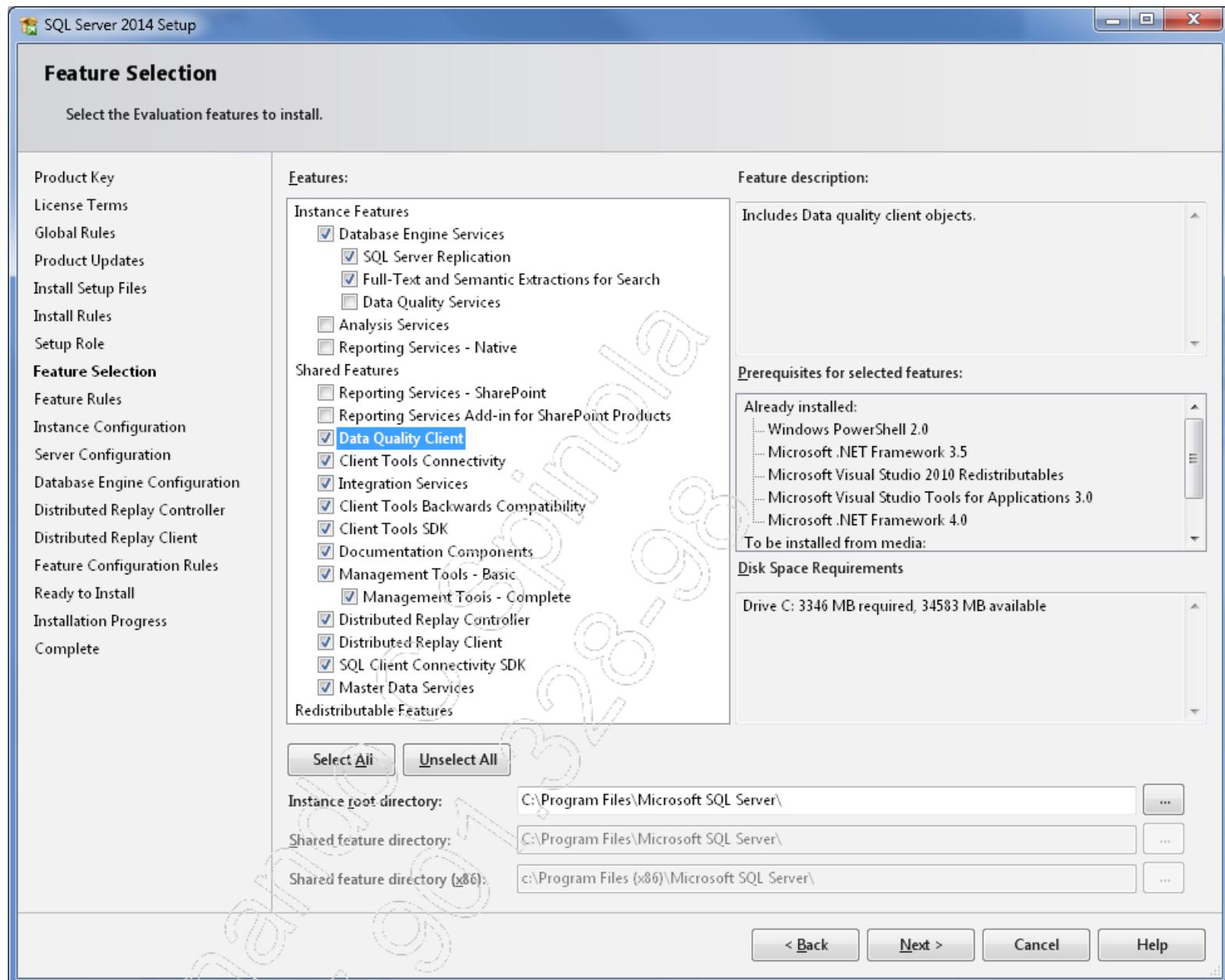


Devemos selecionar o tipo de instalação desejada (somente SQL Server, SQL Server e integração com SharePoint ou todos os recursos configurados de forma padrão).

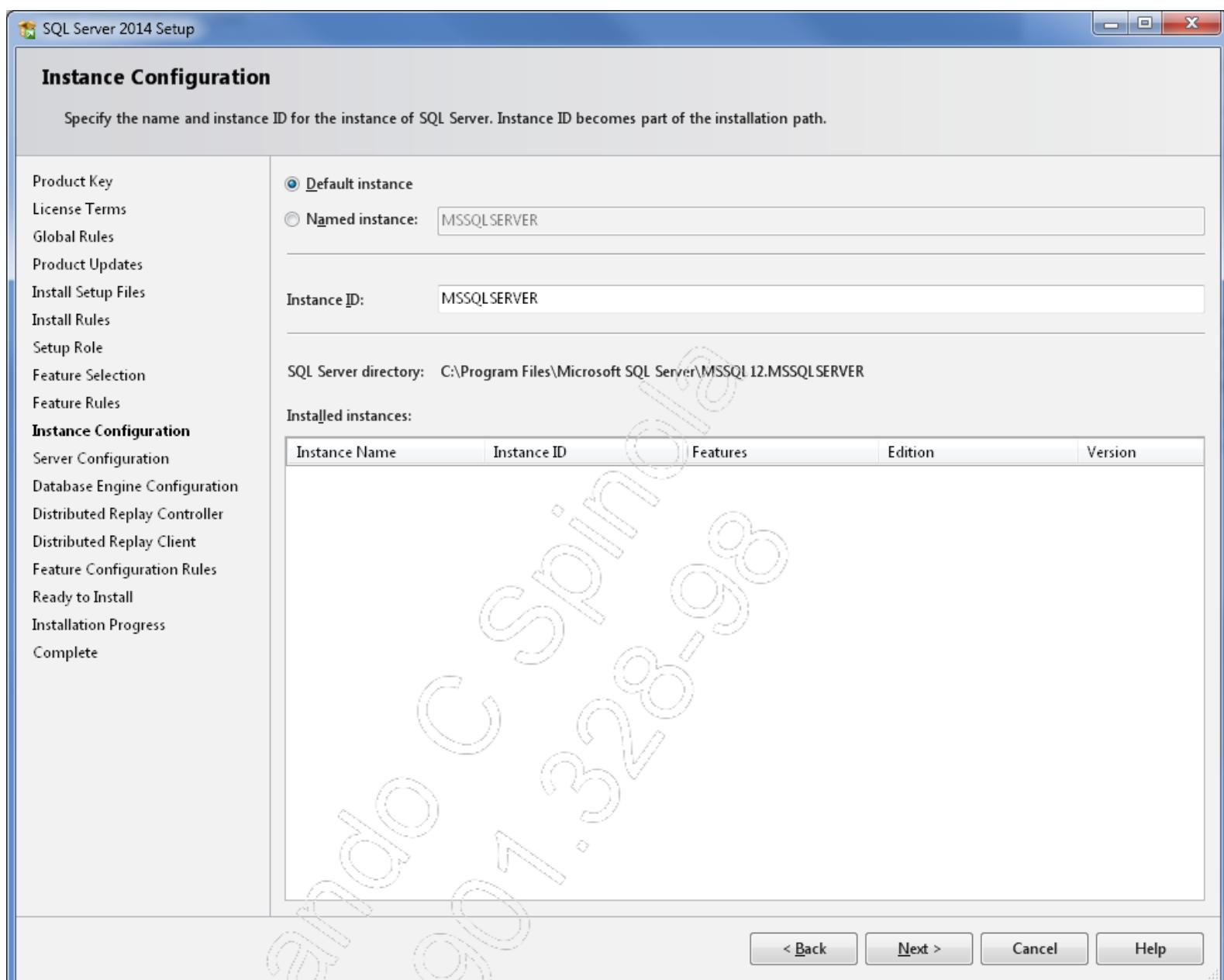


SQL 2014 - Módulo III

Precisamos selecionar os recursos a serem instalados e indicar o diretório de instalação do SQL Server.



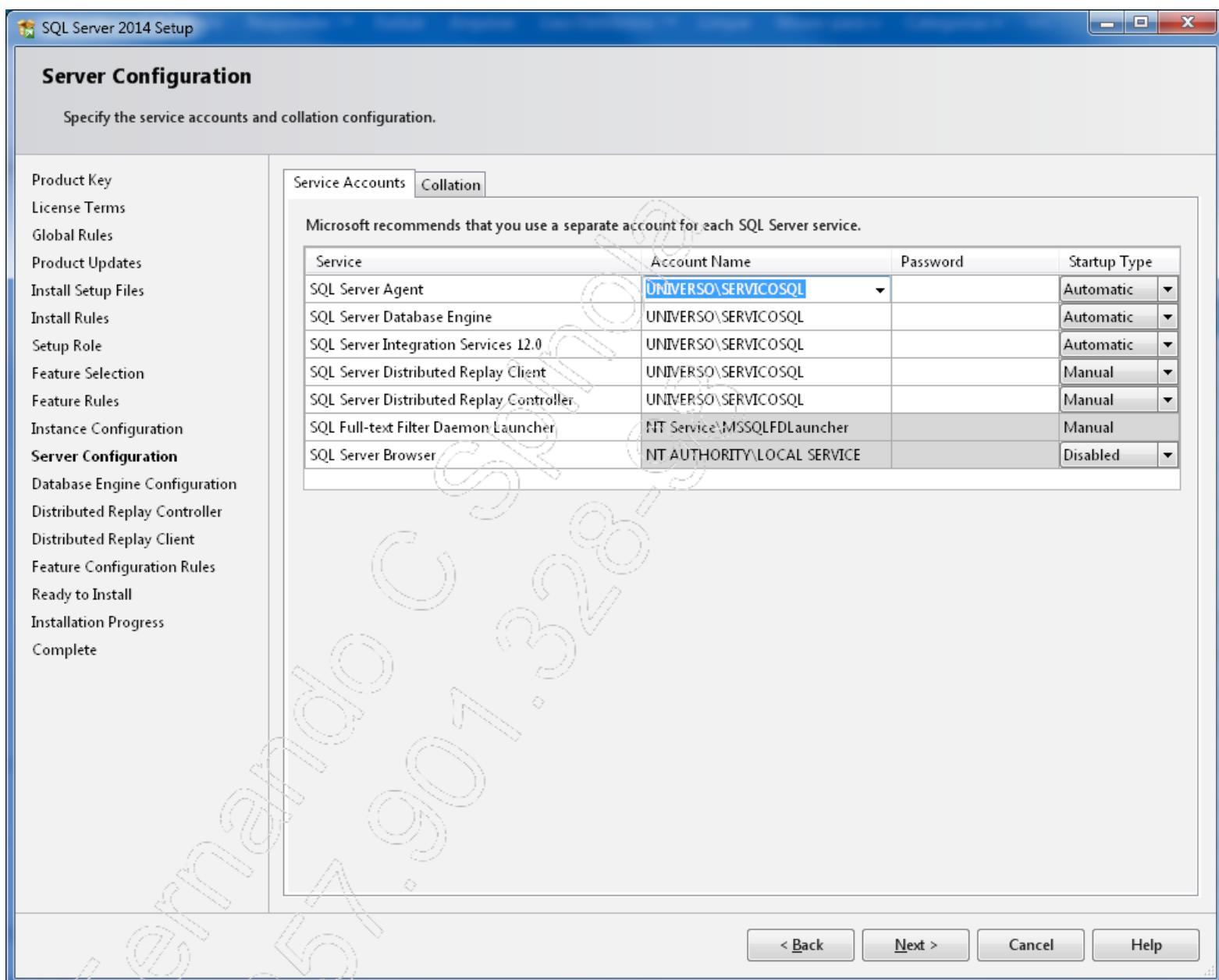
Clique em **Next**.



Neste passo da instalação, escolhemos o nome da instância. Existem duas formas de fazer isso: uma delas é selecionar a instância default (**Default Instance**). Nesse caso, o nome da instância será um nome padrão (**MSSQLSERVER**). A outra forma é selecionar a instância nomeada (**Named Instance**) e então selecionar o nome da instância. Caso um servidor SQL Server seja projetado para múltiplas instâncias, o mais apropriado é utilizar apenas instâncias nomeadas. Caso tenha apenas uma única instância, você pode optar pelo uso da instância Default. Agora, selecione o próximo passo (**Next**).

SQL 2014 - Módulo III

Nesta etapa, definimos para cada serviço do SQL Server o usuário responsável (**Account Name**) pela inicialização e encerramento dos serviços. O detalhamento dos serviços será tratado mais adiante.



Podemos definir como responsável **NT AUTHORITY\NETWORK SERVICES**, o que indica que qualquer usuário que acessar esse servidor poderá realizar as operações mesmo não tendo privilégios de administradores.

- **Conta de usuários do domínio**

São usuários criados em uma rede associada a um determinado domínio, caso o serviço do Windows tenha a necessidade de interagir com os serviços de rede, de compartilhar arquivos ou ainda de usar alguma conexão vinculada a outros servidores SQL Server. O ideal é que esses usuários tenham o mínimo possível de privilégios. Cabe ao administrador do ambiente do sistema operacional a criação do arquivo.

- **Conta Usuário local**

Caso o servidor não faça parte de uma rede com domínio, uma conta de usuário deverá ser criada no próprio servidor. Por questões de segurança, esta conta não deve ter permissões de administrador. Caso não tenha uma rede local com domínio, esta deve ser a forma de inicialização dos serviços.

- **Conta Serviço local**

Este tipo de conta tem um caráter interno e possui uma equivalência com o grupo chamado de usuários do Windows (**Users**). Além disso, essa conta não possui suporte quando é associada aos serviços de instância e do SQL Server Agent. O nome dessa conta é **NT AUTHORITY\LOCAL SERVICE**.

- **Conta Serviço de rede**

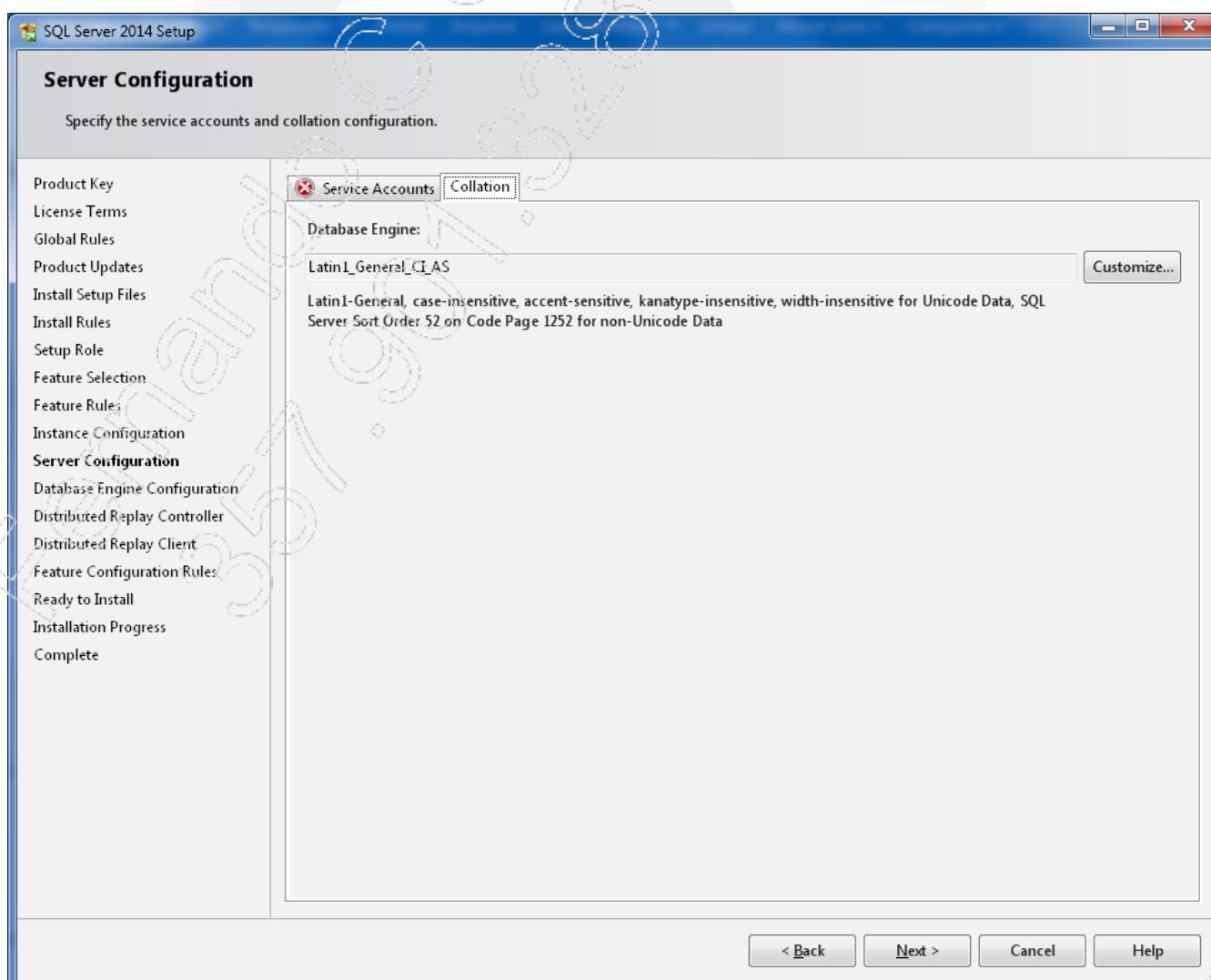
Este tipo de conta tem mais acesso a recursos, arquivos e diretórios que o grupo de usuários (**Users**). Os serviços executados com esta conta acessam recursos de rede por meio das credenciais da conta conectada ao servidor. Esta conta é denominada **NT AUTHORITY\NETWORK SERVICE**. Devemos evitar esta opção a todo custo, pois ela oferece grandes riscos de segurança.

- **Conta Sistema local**

Conta interna que possui um alto grau de privilégios, não recomendada de forma alguma para autenticar os serviços do SQL Server. Esta conta é conhecida como **NT AUTHORITY\SYSTEM**.

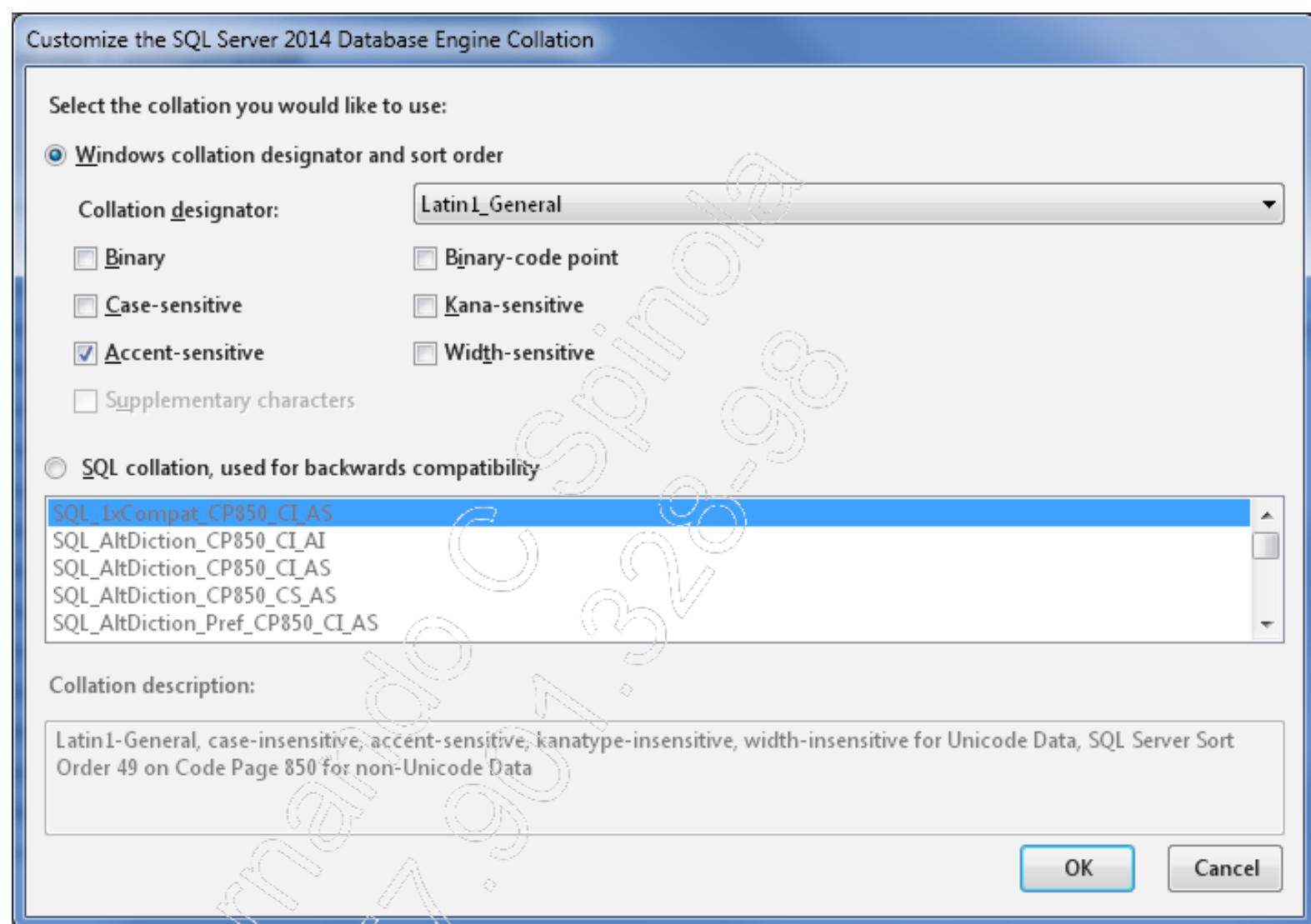
Selecione a aba **Collation**. Agora, selecione o próximo passo (**Next**). Nesta etapa, definiremos o collation da instalação SQL Server, que pode ser customizada. O collation para o SQL Server tem uma série de objetivos, por exemplo:

- Definir o tipo de língua utilizada no armazenamento de dados;
- Definir se as buscas levarão em conta o uso de maiúsculas e minúsculas (Case Sensitive - CS ou Case Insensitive - CI);
- Definir a ordenação de caracteres acentuados (Accent Sensitive – AS ou Accent Insensitive – AI);
- Base do collation para criação de novos databases.



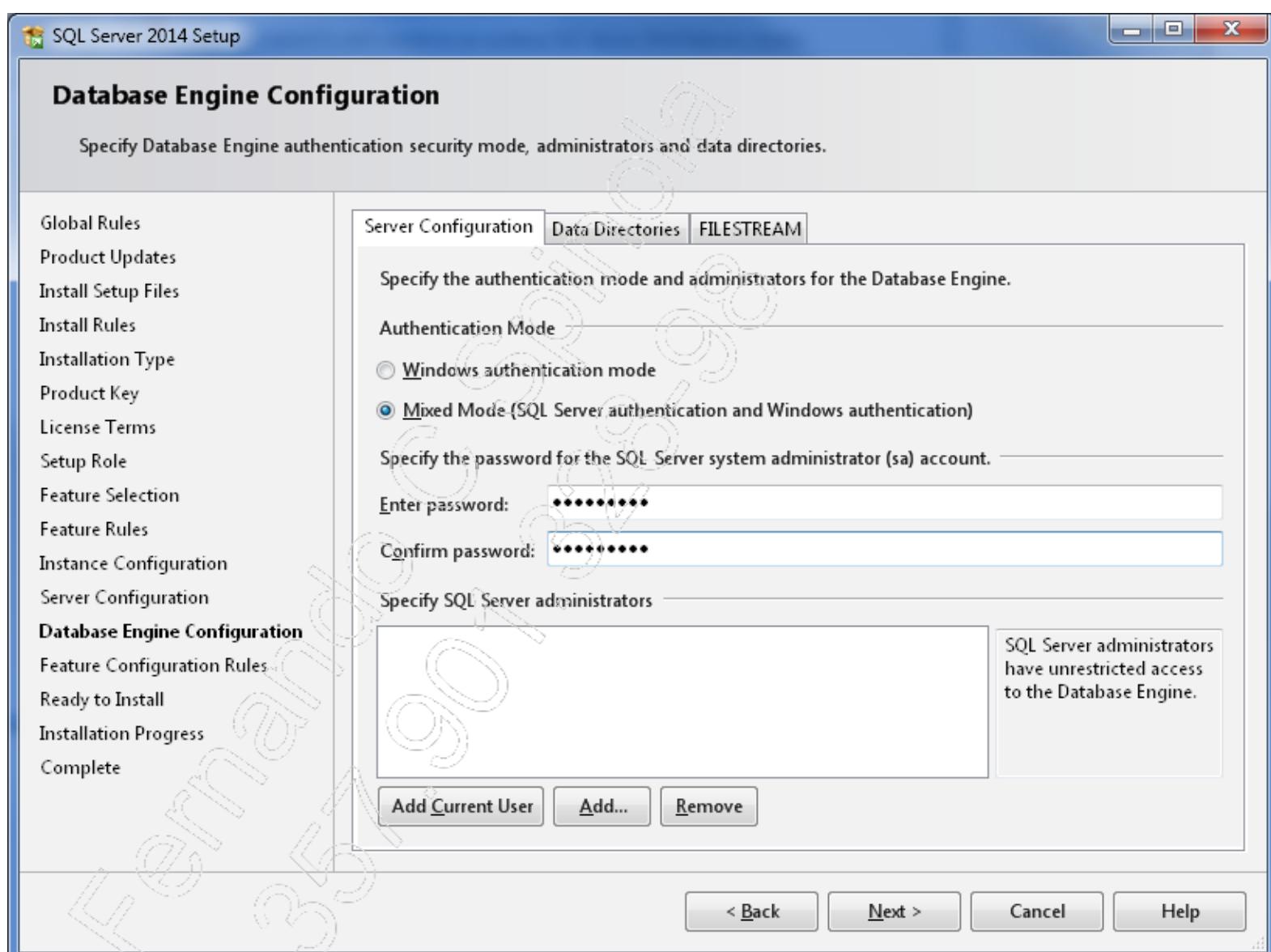
O **latin1** é um collation que pode ser usado para todas as línguas latinas e suporta os acentos dessas línguas.

Se quisermos, podemos customizar o collation utilizando o botão **Customize**. Podemos escolher entre **Windows Collation** e **SQL Collation** (usado para compatibilidade com versões anteriores).



SQL 2014 - Módulo III

Clique em **Cancel** e depois em **Next**. A tela **Database Engine Configuration** será exibida. Neste passo, devemos selecionar o padrão de autenticação a ser utilizado pela instância SQL Server. Após a instalação, esta opção poderá ser modificada, caso necessário. Optar pelo Windows permite que a autenticação para entrada na instância seja feita por esse sistema operacional, pois, se o usuário no Windows for autenticado e o mesmo usuário tiver sido criado no SQL Server, ele estará autenticado no SQL Server sem o uso de senhas adicionais.



Usando o modo misto (**Mixed Mode**), podemos contar com usuários autenticados via sistema operacional ou também com usuários autenticados pelo SQL Server, logo, a senha é administrada pelo SQL Server.

Neste momento, podemos indicar que o usuário instalador também será o usuário administrador através do botão **Add Current User**. Podemos adicionar quantos usuários quisermos usando o botão **Add....**

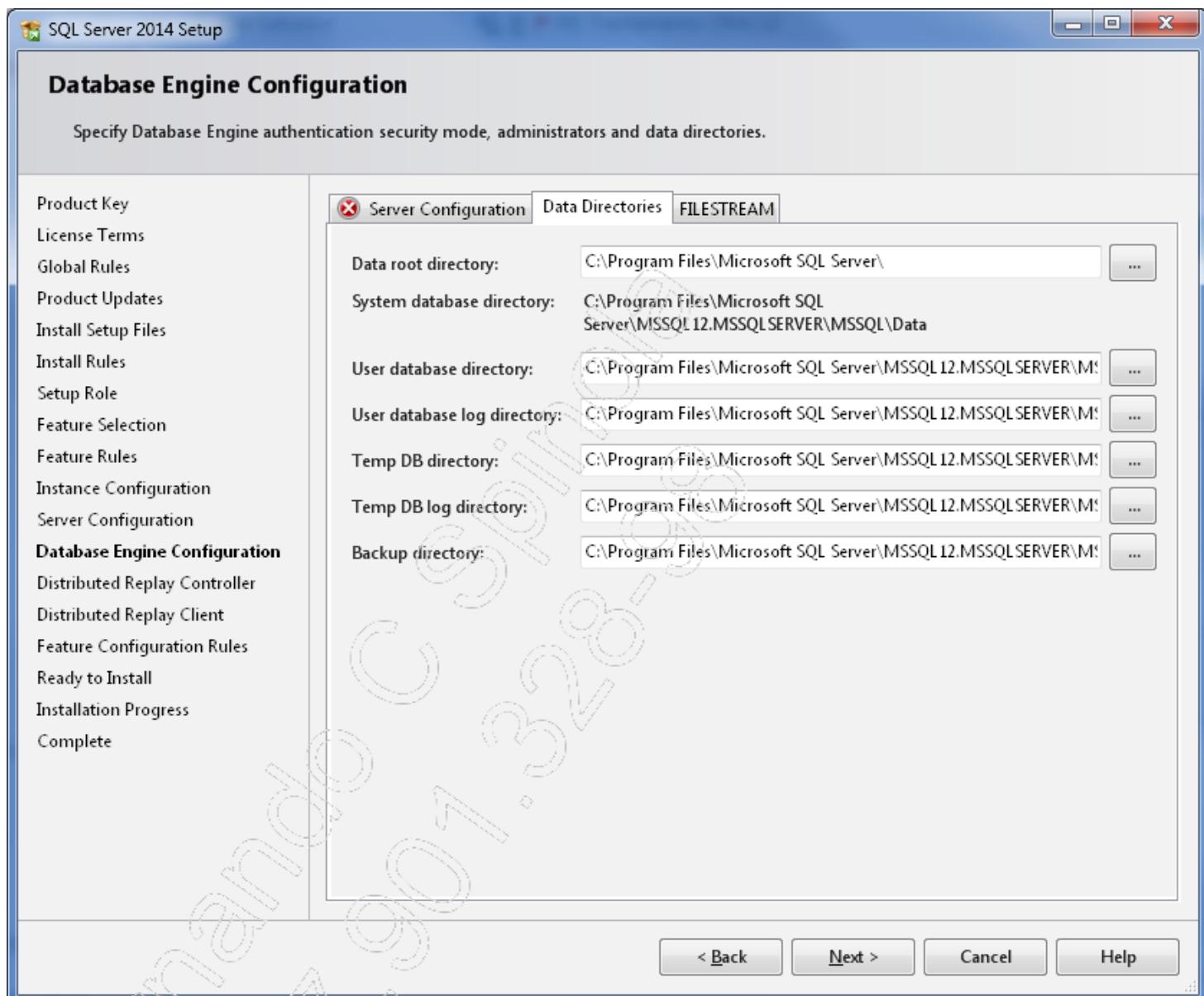
Se escolhermos a opção **Mixed User**, será criado um usuário chamado **sa** (system administrator) que será também um usuário administrador da instância. Neste momento deveremos definir a senha para esse usuário. Esta senha será validada conforme os critérios de senha usados no sistema operacional. Geralmente oito ou mais caracteres, sendo pelo menos um número e 1 caractere especial (#@!\$%&*?+).

Para a instalação do SQL no ambiente da Impacta, será necessário realizar os seguintes passos:

- Selecione a opção **Mixed Mode** (autenticação do SQL e do Windows);
- Informe a senha, por exemplo: Imp@ct@12;
- Clique no botão **Add Current User** para adicionar a conta do usuário como administrador;
- Adicione a conta do domínio: **UNIVERSO\SERVICOSQL** como administrador.

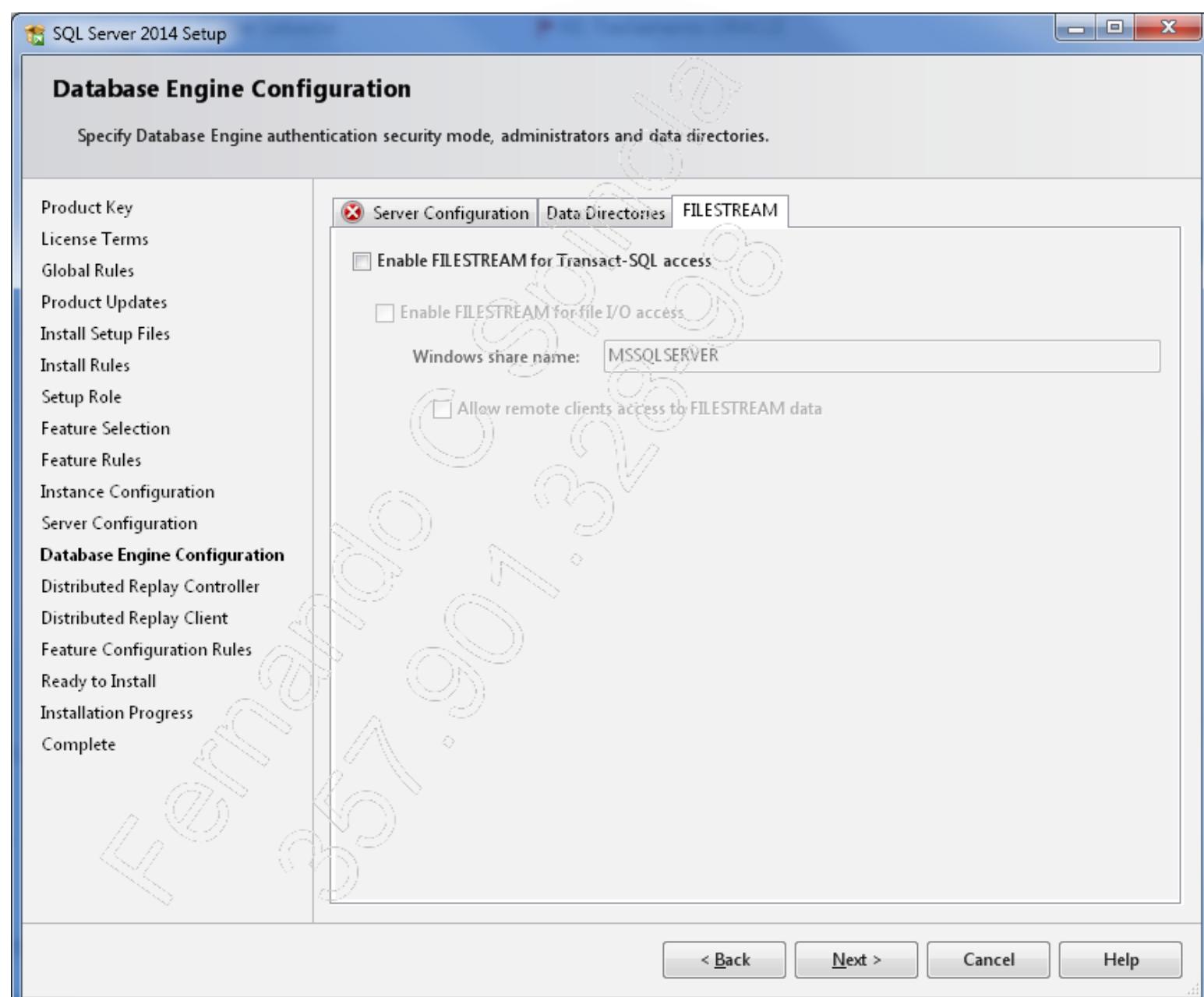
SQL 2014 - Módulo III

Selecione a aba **Data Directories**. Nesta opção, defina os diretórios nos quais serão armazenados os dados. Recomenda-se que sejam separados do diretório da instalação, preferencialmente em discos diferentes (se possível).



Clique em **Next**.

O **FILESTREAM** é um recurso do SQL Server que possibilita que aplicativos armazenem dados não estruturados, como imagens, no sistema de arquivos. Embora o banco de dados SQL Server possa armazenar dados binários, este recurso pode consumir muito processamento e espaço no banco de dados, sem contar a performance que é inferior ao que podemos obter com os dados não estruturados armazenados em disco. Através de APIs de streaming e do sistema de arquivos, é possível manter a consistência dos dados transacionais e dos dados não estruturados armazenados utilizando este recurso.

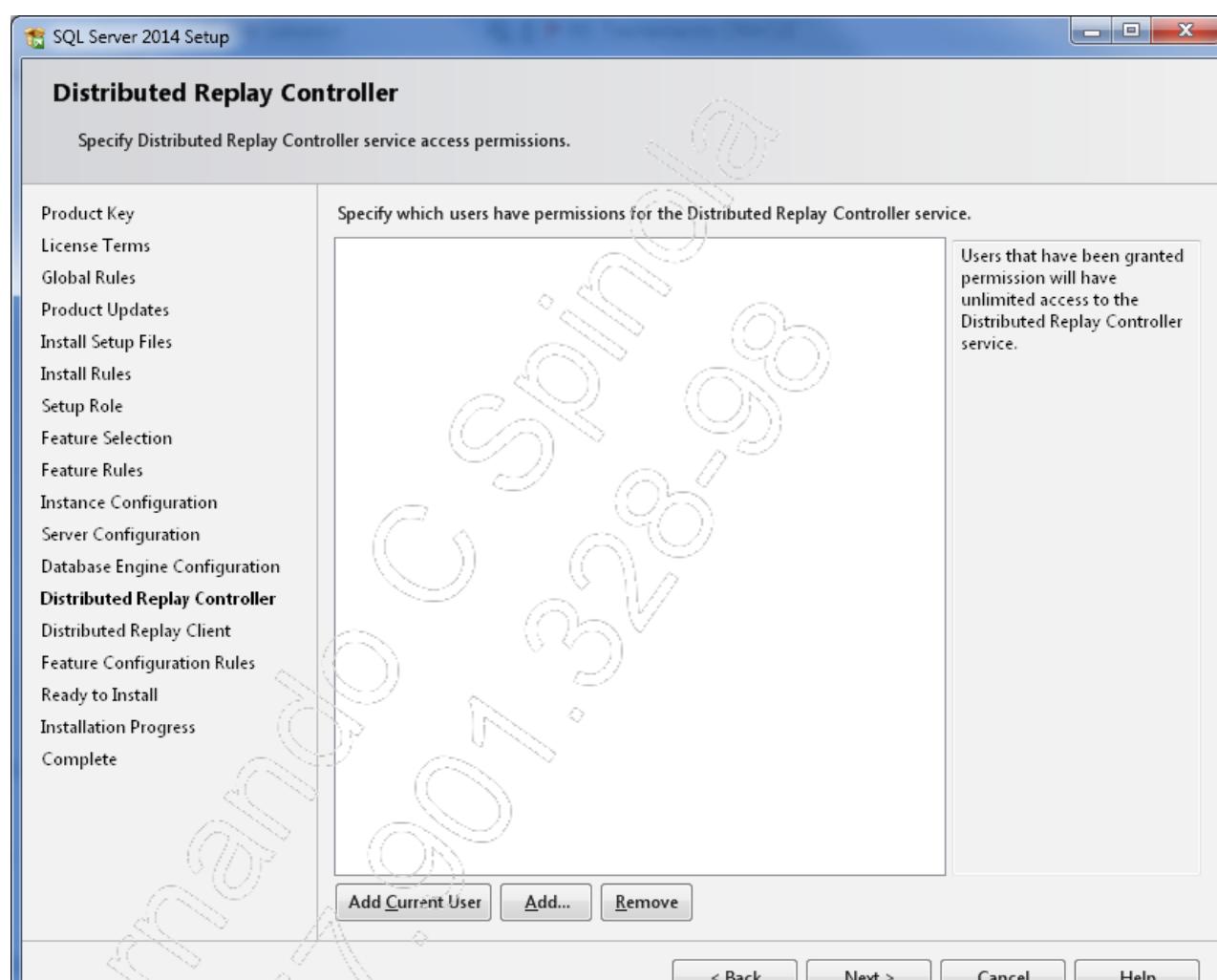


Esse recurso de **FILESTREAM** integra o banco de dados com o sistema de arquivos do Windows, armazenando dados em tipos de dados **BLOB** (Binary Large Object) e **varbinary** no sistema de arquivos do SQL Server. Comandos DML e instruções **transact** podem ser utilizados com **FILESTREAM**.

Além disso, esse recurso realiza o cache de arquivos do Windows para armazenar os dados recuperados de um arquivo texto, o que auxilia a reduzir efeitos negativos de uso do recurso. Nenhuma das estruturas da instância SQL Server é utilizada pelo **FILESTREAM**, por outro lado isso permite que os dados estruturados sejam carregados pelo conjunto de buffers da instância SQL Server, enquanto os dados não estruturados são carregados também em memória, mas no sistema operacional.

Quando o SQL Server é instalado, o **FILESTREAM** não é habilitado. Para habilitá-lo, devemos usar o **SQL Server Configuration Manager** e o **SSMS**. Para que possamos usar esse recurso, a tabela deverá ter uma coluna com tipo de dados **varbinary(max)** definida com a opção **FILESTREAM**. Clique em **Next**.

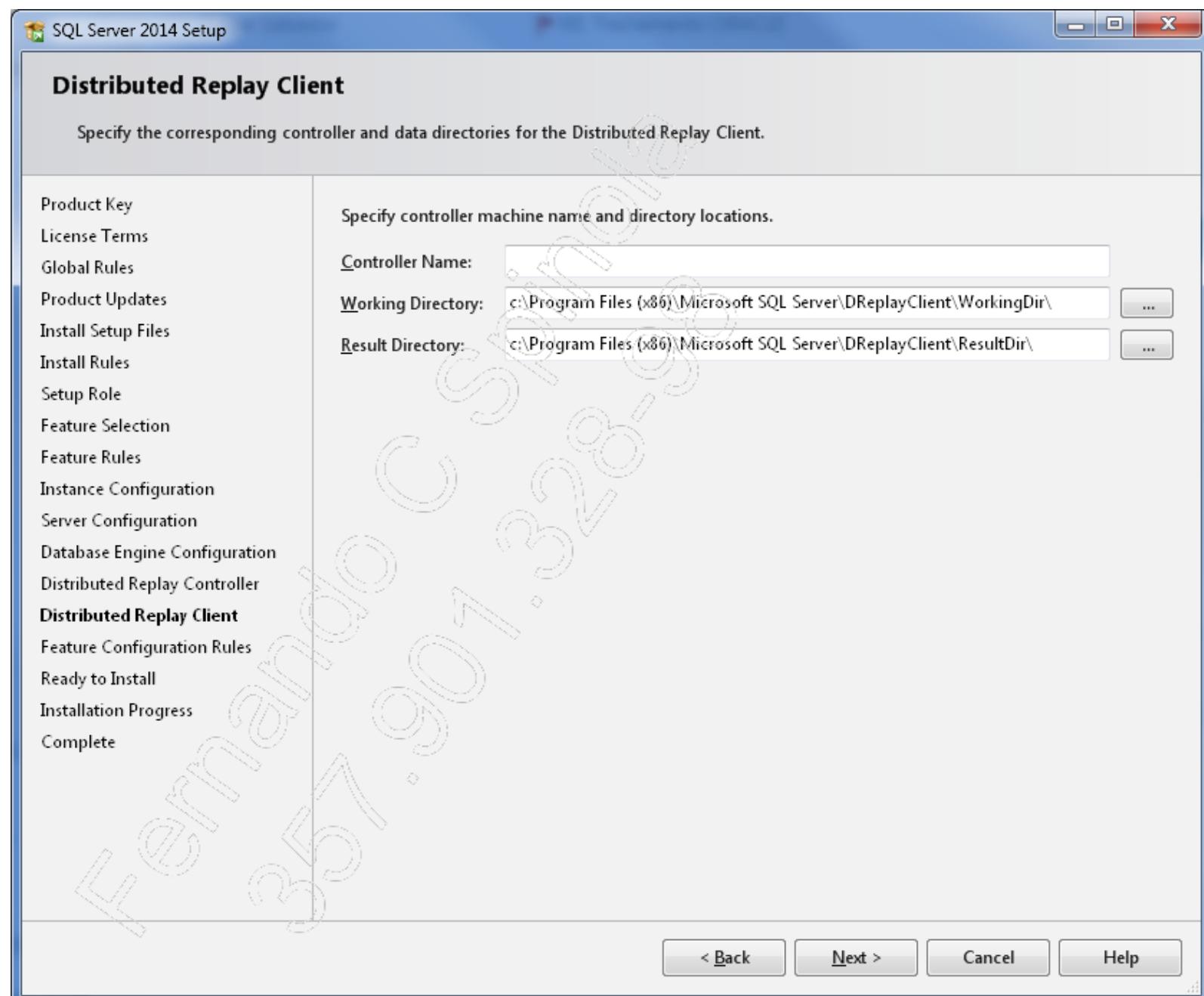
O **Distributed Replay Controller** é um recurso introduzido no SQL Server 2014 que possibilita a captura de transações realizadas em um ou mais ambientes e a reprodução destas em um servidor SQL Server, de forma a medir a capacidade do servidor para o atendimento de transações. Isto é muito útil para fazer ou determinar a necessidade de upgrades de hardware em função da carga de trabalho e do respectivo aumento dessa carga.



SQL 2014 - Módulo III

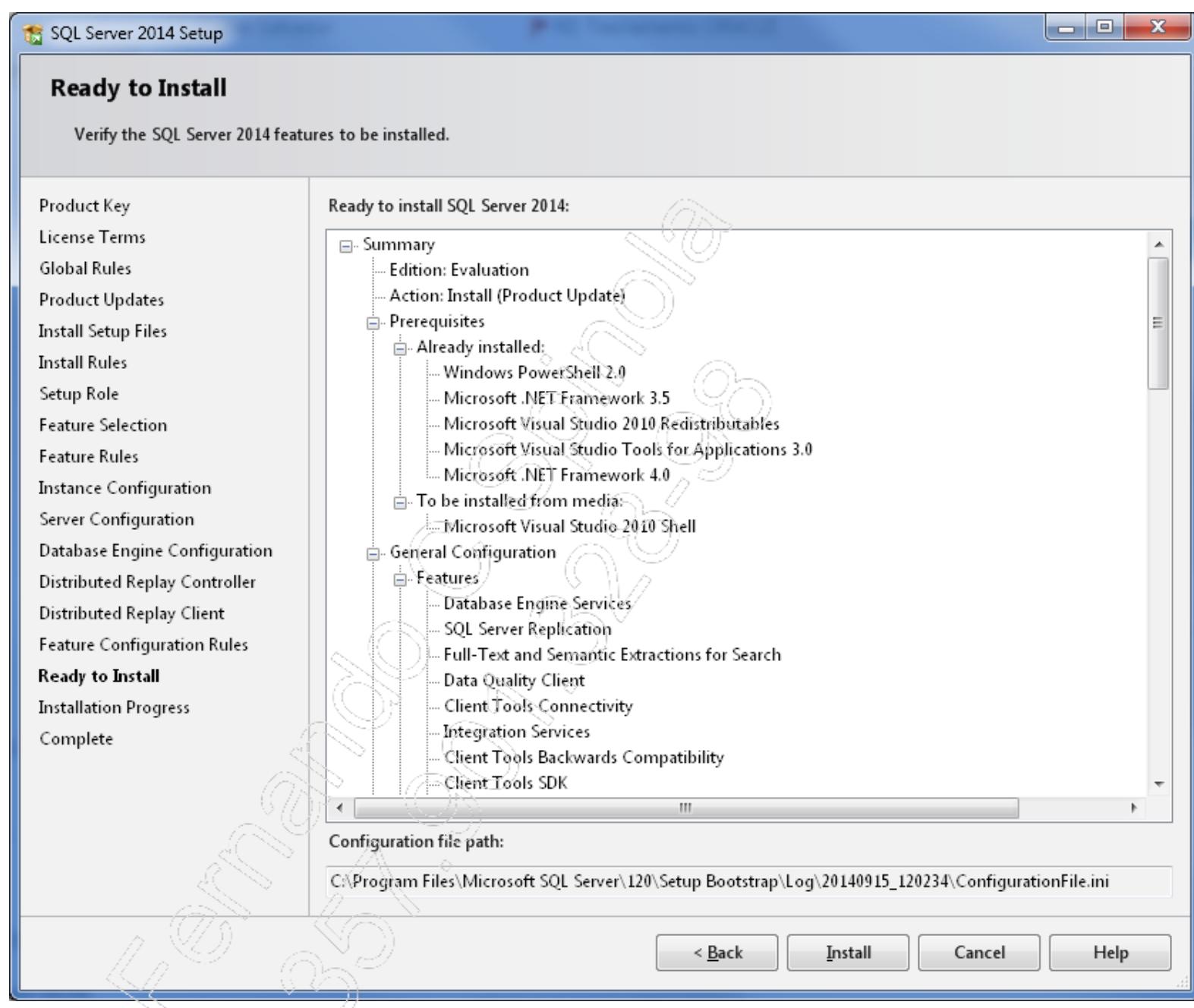
Use o **Distributed Replay** para teste de compatibilidade de aplicativo, teste de desempenho ou planejamento de capacidade.

Cientes do recurso **Distributed Replay** podem atuar em conjunto para simulação de carga de ambiente (seja produção ou homologação de banco de dados). Essa carga é capturada em um ambiente e podemos simulá-la em outro ambiente. Para realizar esta operação, pode-se ter um ou mais clientes em cada ambiente.



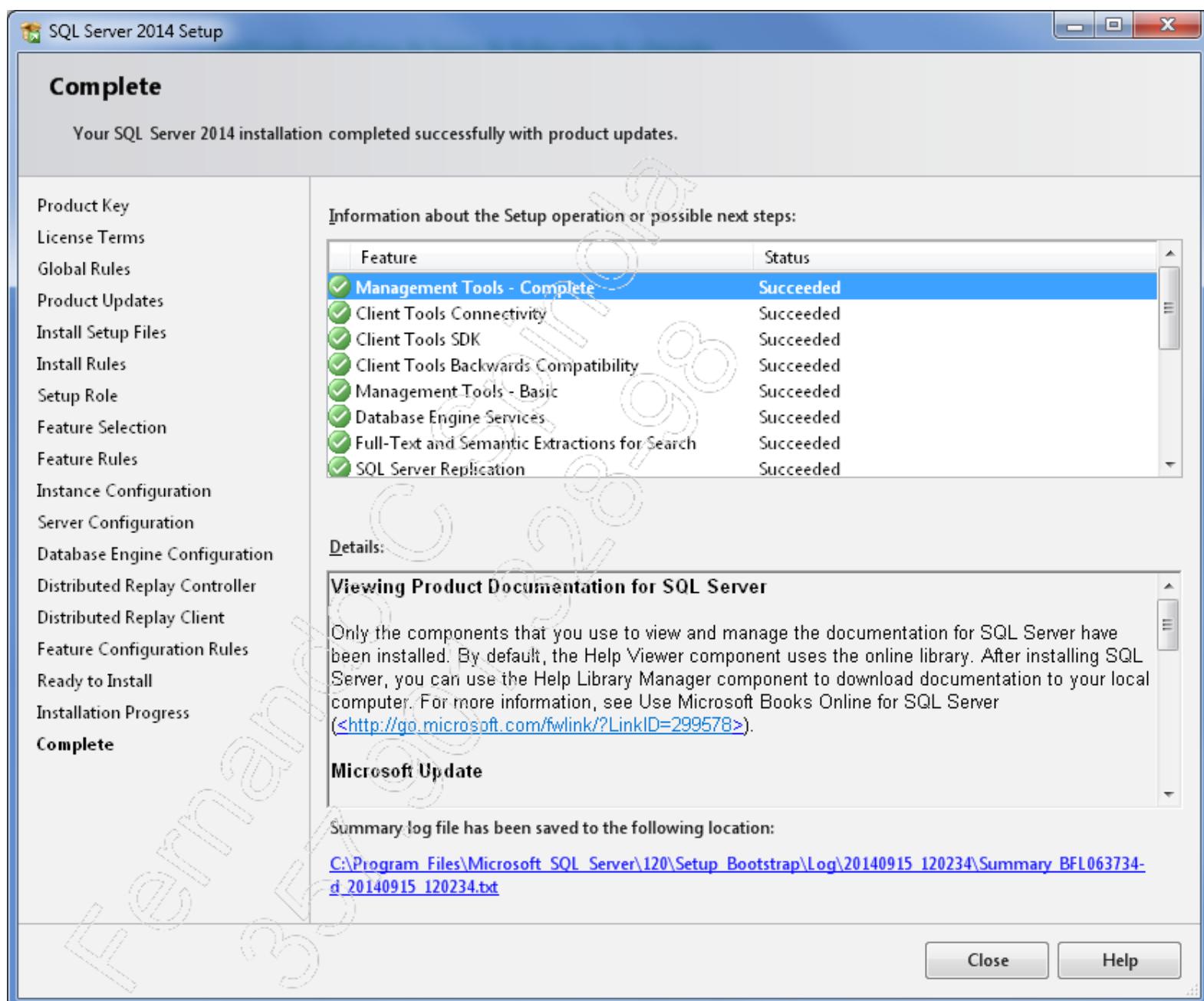
Opcionalmente, o **Error Reporting** pode ser indicado para envio de comportamentos indevidos por parte do SQL Server para a Microsoft.

Esta é a etapa final antes da instalação. Todos os nomes e usuários estão validados. Selecione o próximo passo (**Next**).



SQL 2014 - Módulo III

A instalação será iniciada e, após o término, a tela final da instalação será exibida conforme a imagem a seguir:



Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo.

- A instalação deve ser planejada antes de ser iniciada, logo é importante definir o nome da instância, os padrões usados para o collation, decidir se será utilizado o recurso de FILESTREAM e o recurso de distributed replay;
- Definir a forma como o login será realizado na instância também é importante, porém, podemos alterar esta opção, caso necessário;
- Analysis Services e Reporting Services são recursos opcionais na instalação do SQL Server 2014;
- A instalação do SQL Server 2014 consiste em duas partes: os serviços ligados à instância e os recursos compartilhados do SQL Server;
- Um servidor pode ter até 50 instâncias em um ambiente stand-alone e 25 em ambientes clusterizados.

1

Instalando o SQL Server

Teste seus conhecimentos

Fernando Góspoli
357.907.300-0008



IMPACTA
EDITORA

1. Precisamos ter duas instâncias em um mesmo servidor. Com relação a esta afirmação, qual alternativa está correta?

- a) Duas instâncias usando (Default Instance).
- b) Uma instância Default e outra instância nomeada (Named Instance).
- c) Neste caso, obrigatoriamente, instâncias nomeadas (Named Instance).
- d) Não é possível ter duas instâncias no mesmo servidor.
- e) Nenhuma das alternativas anteriores está correta.

2. Qual sistema operacional é suportado pelo SQL Server 2014?

- a) Windows Server 2008 (32 ou 64 Bits) ou superior.
- b) Qualquer Windows.
- c) Windows Server 2014.
- d) Windows Server 2008 e também versões mais novas de Linux.
- e) Somente Windows Server 2012.

3. O que é FILESTREAM?

- a) Recurso de arquivamento de dados.
- b) Recurso que otimiza o armazenamento de objetos binários no repositório de dados.
- c) Recurso de arquivamento.
- d) Recurso de arquivamento de objetos binários no sistema de arquivos do Windows.
- e) Recurso de arquivamento que não deve ser utilizado.

4. Quais são os passos que devemos realizar antes de fazer a instalação do SQL Server 2014?

- a) Cópia dos binários para o disco, criação de usuário para serviços, definição de discos e permissões.
- b) Verificação da fragmentação dos discos, criação de usuário para serviços, definição de discos e permissões.
- c) Criação de usuário para serviços, definição de discos e permissões.
- d) Verificação da fragmentação dos discos, definição de discos e permissões.
- e) Verificação da fragmentação dos discos, cópia do Regedit, criação de ponto de restauração, cópia dos binários para o disco, criação de usuário para serviços, definição de discos e permissões.

5. Qual recurso foi incorporado a partir da versão SQL Server 2012?

- a) Particionamento de tabelas.
- b) FILESTREAM.
- c) Distributed Replay Controller.
- d) Auditoria.
- e) Reporting Services.

1

Instalando o SQL Server

Mãos à obra!

Fernando Spinola
357.907.

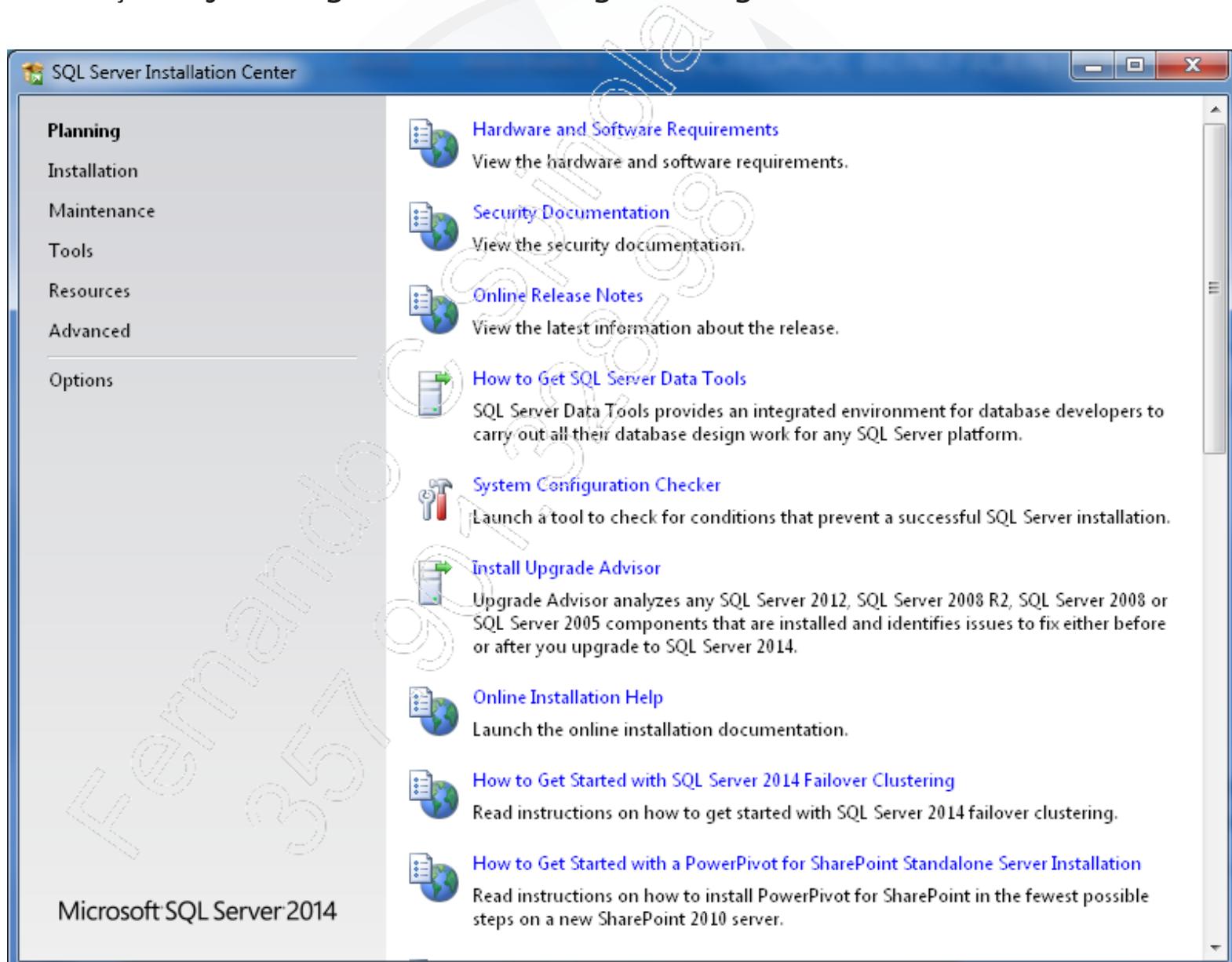


IMPACTA
EDITORA

Laboratório 1

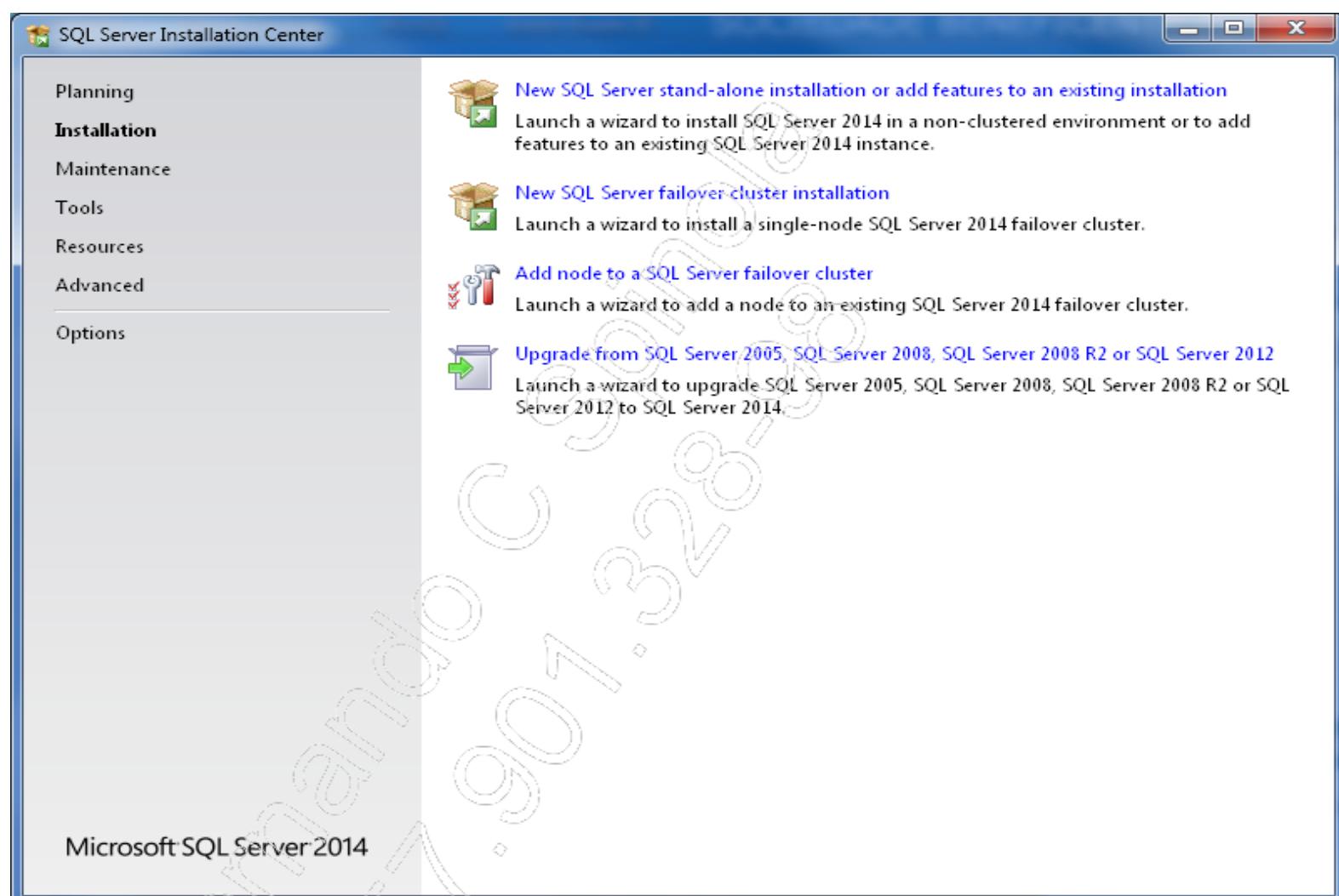
A – Instalando a instância default do SQL Server 2014

1. No **Windows Explorer**, expanda a pasta C:\SQLServer2014 - Módulo III;
2. Entre na pasta **SQLServer2014**;
3. Efetue um duplo-clique sobre o arquivo **SETUP** e aguarde até que a tela de instalação seja carregada como a imagem a seguir:



4. Clique na opção **Installation** na guia à esquerda;

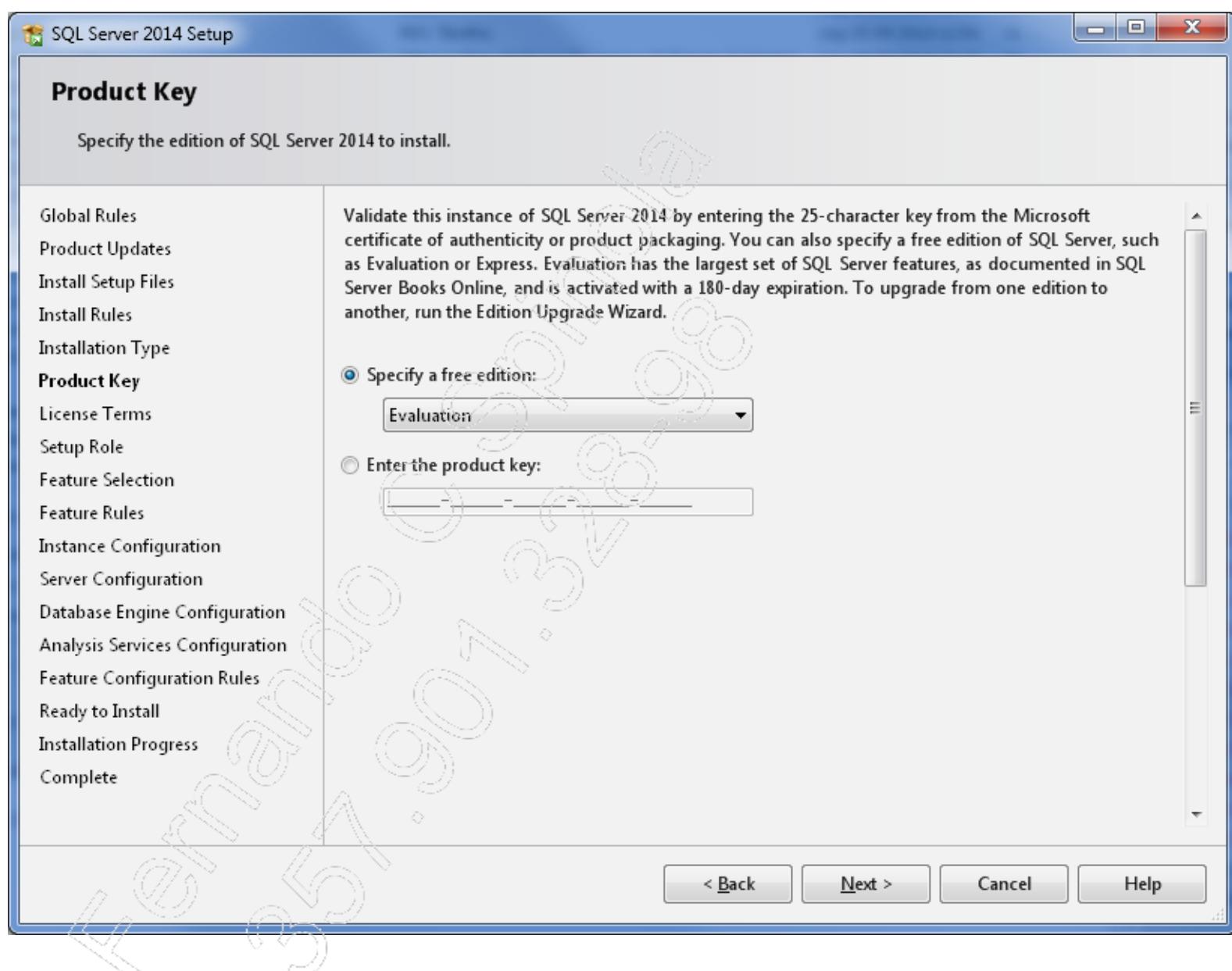
5. Escolha a opção **New SQL Server stand-alone installation or add features to an existing installation;**



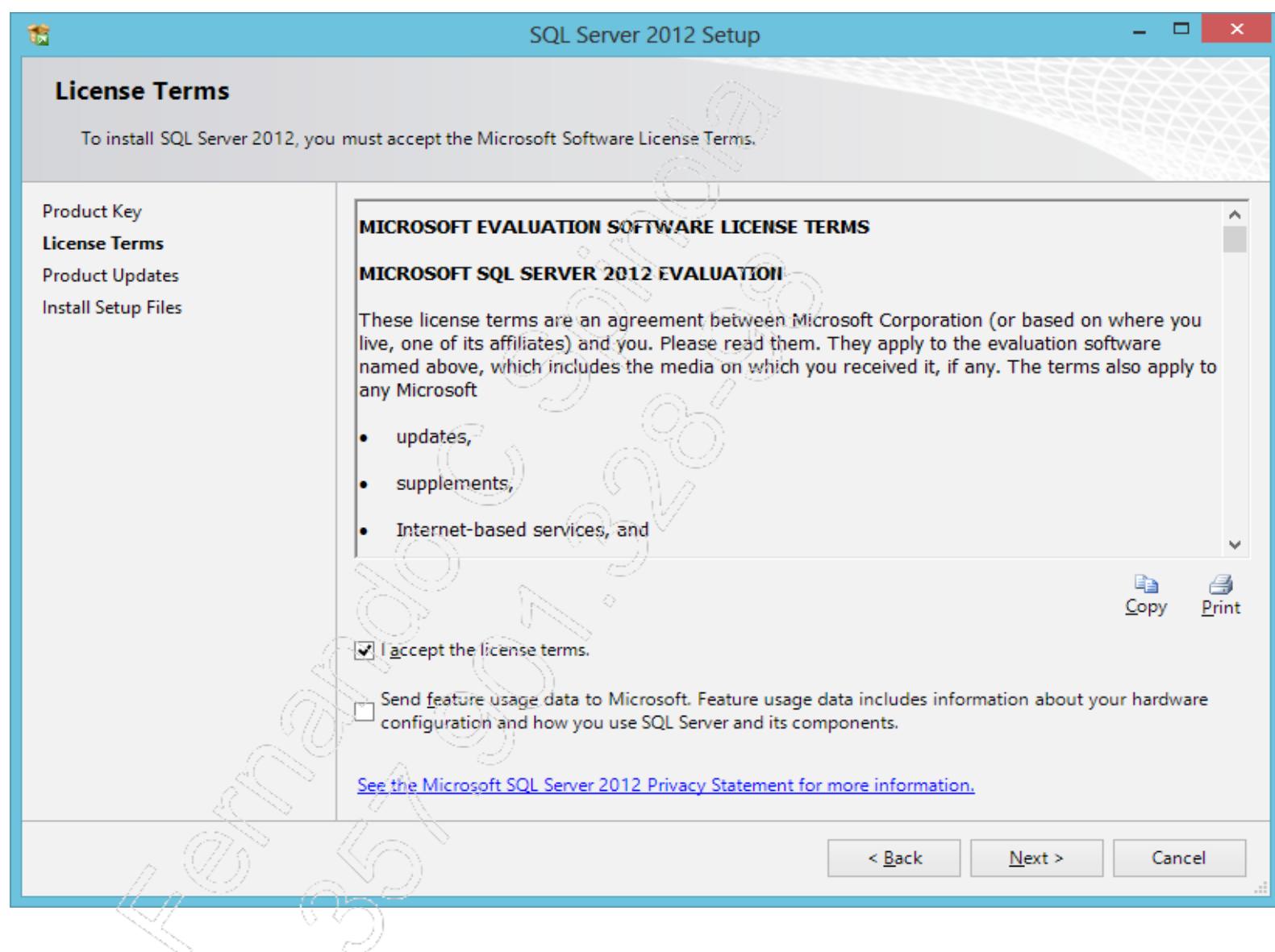
6. Pressione **Ok** na tela **Setup Support Rules**;

SQL 2014 - Módulo III

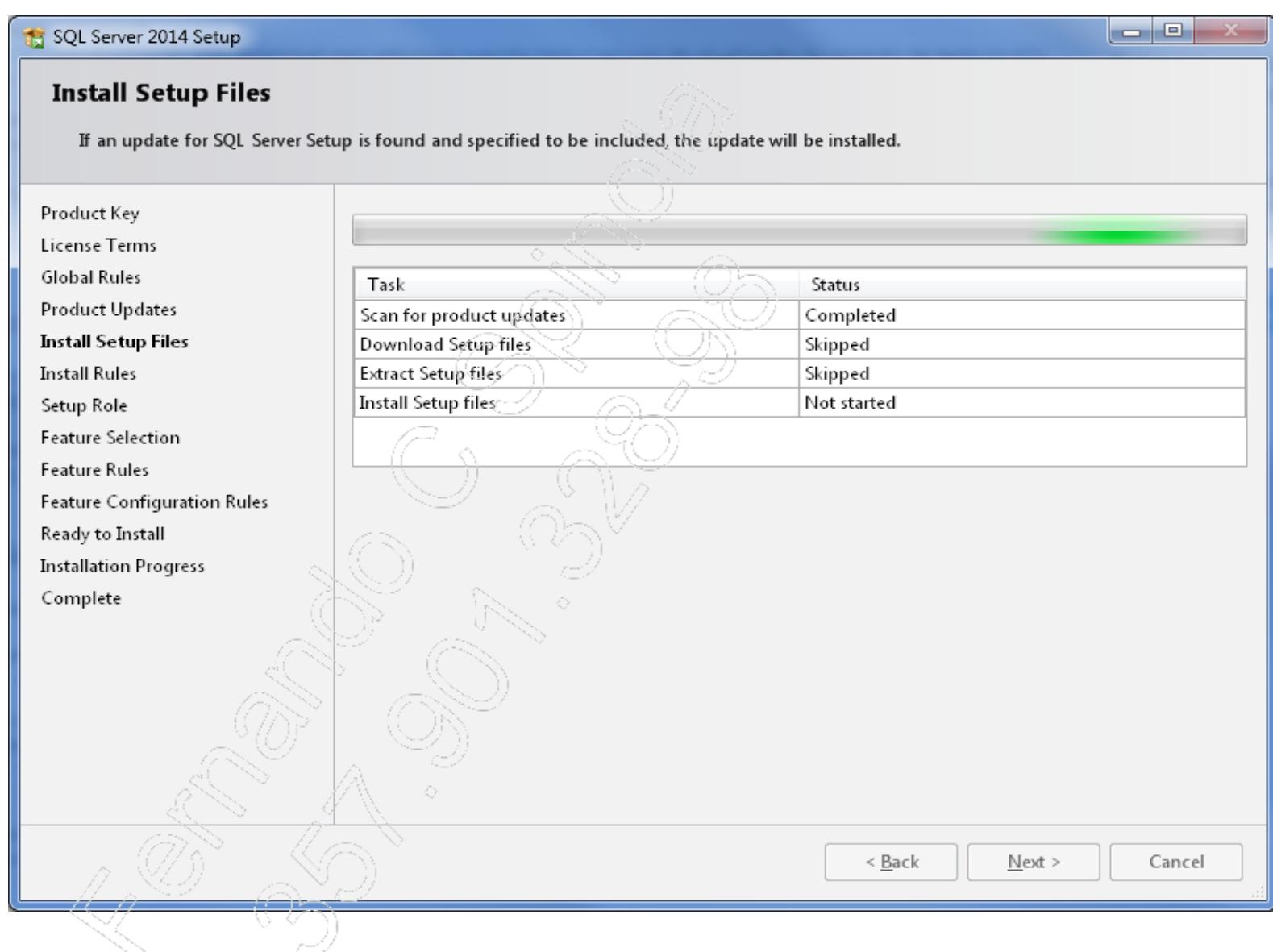
7. Especifique a licença Enterprise Evaluation e clique em Next;



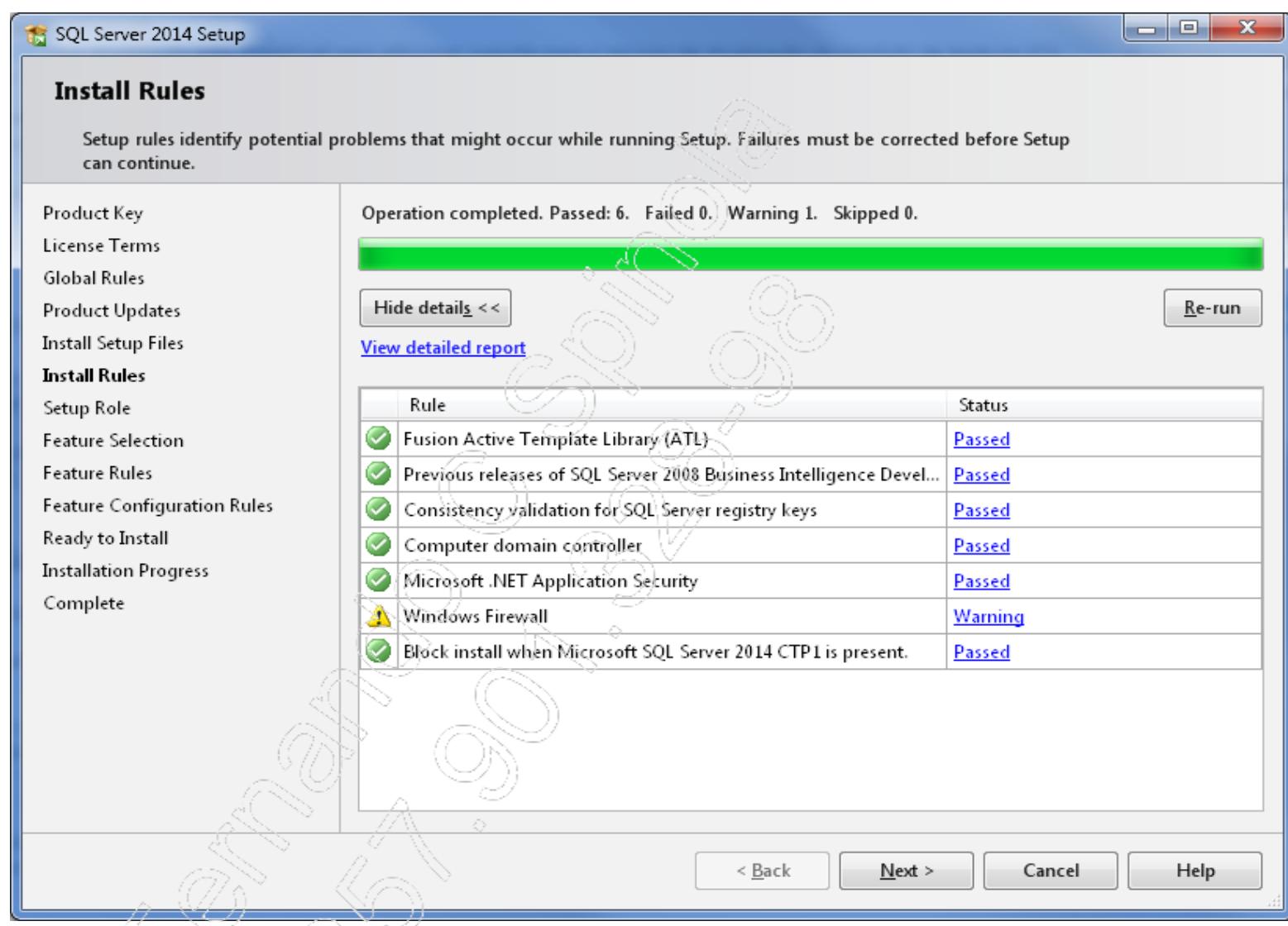
8. Efetue a leitura dos termos de licença, selecione a opção **I accept the license terms** e clique em **Next**;



9. Selecione a opção **Include SQL Server product updates** para incluir as últimas atualizações;

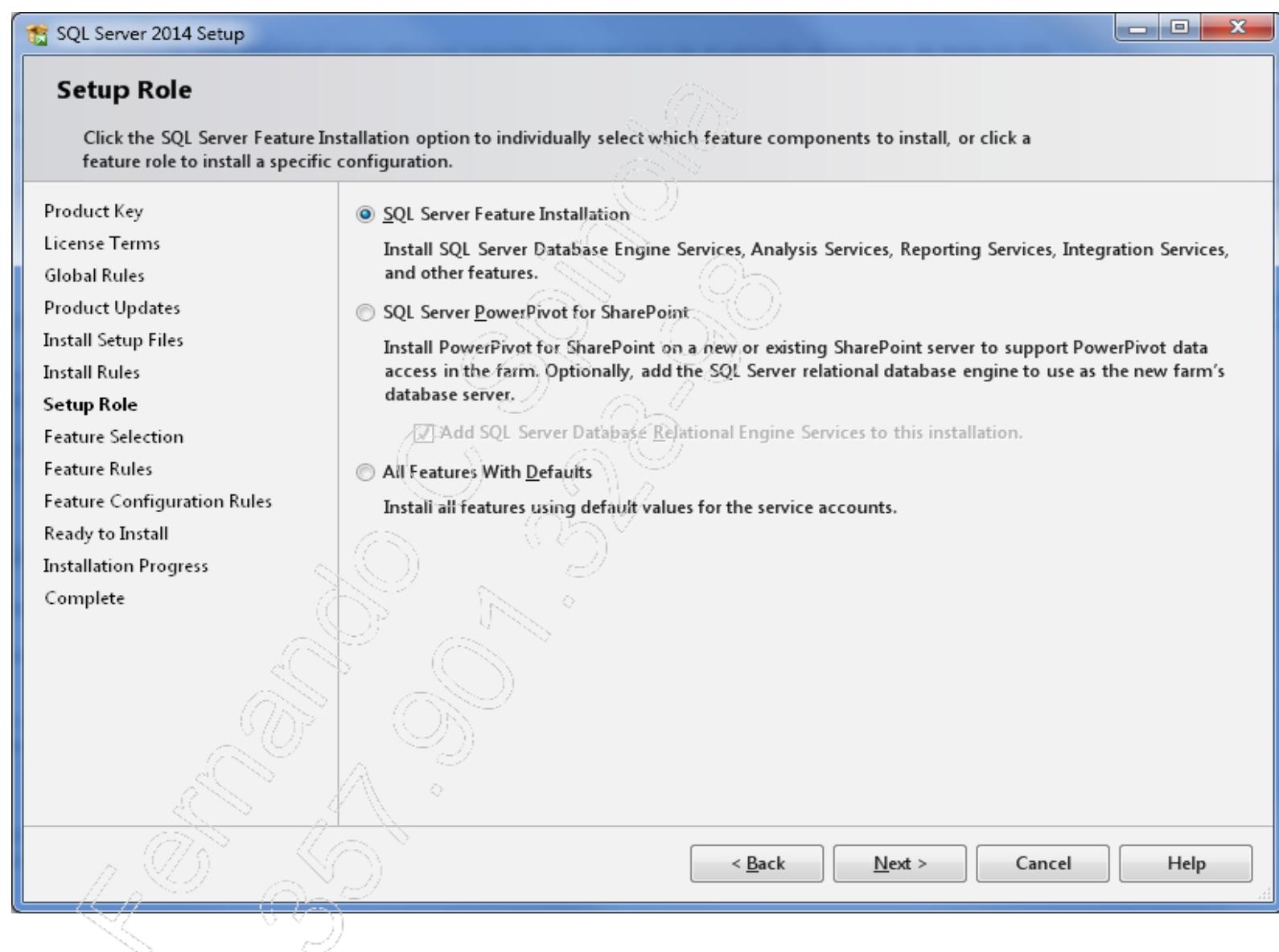


10. Clique em Next na tela Setup Support Rules;



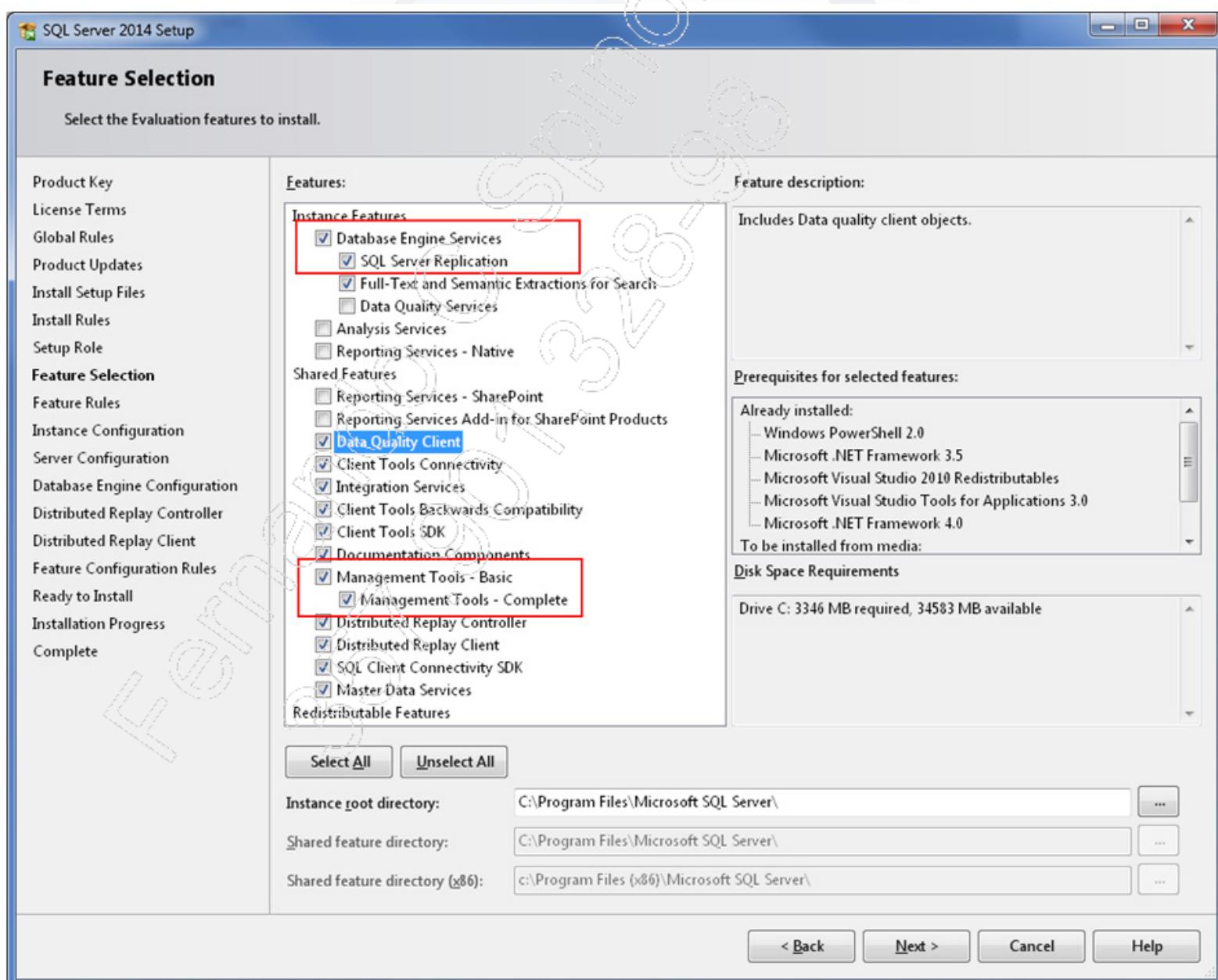
SQL 2014 - Módulo III

11. Selecione a opção **SQL Server Feature Installation** para escolher a opção de instalação e clique em **Next**:



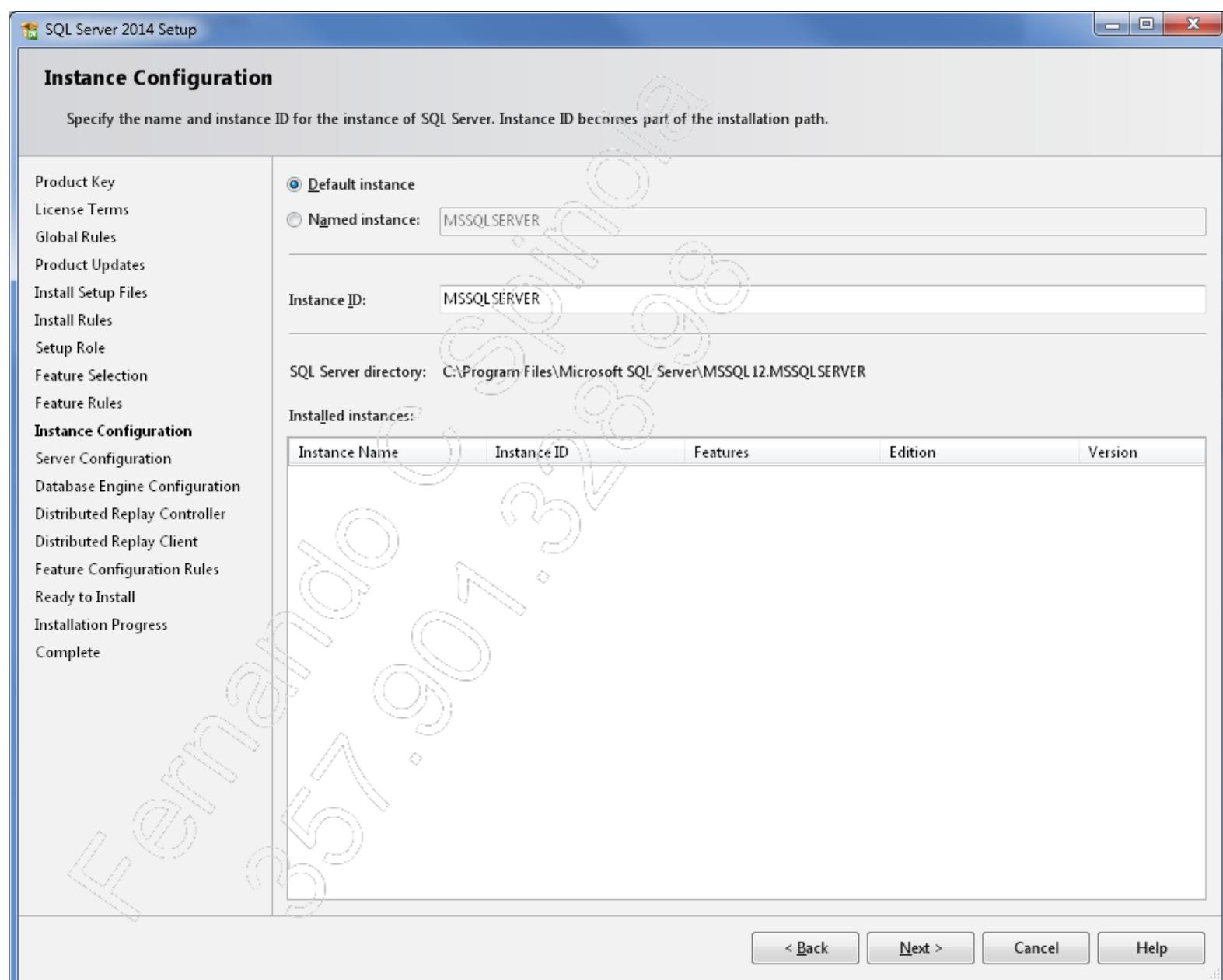
12. Na tela de Seleção de Opções (Feature Selection), selecione as opções a seguir:

- Instance Features
 - Database Engine Services
 - SQL Server Replication
- Management Tools – Basic
 - Management Tools – Complete



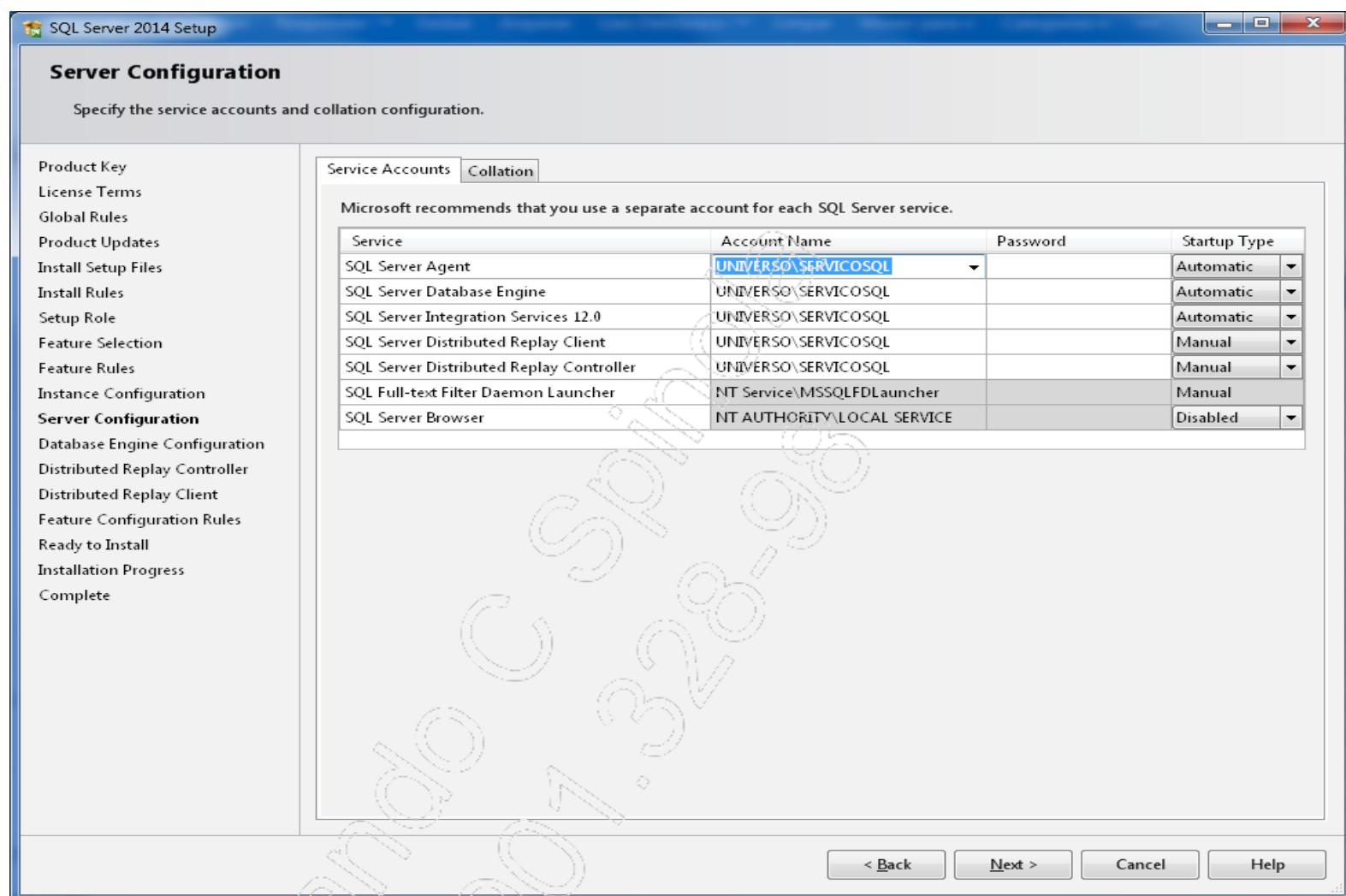
SQL 2014 - Módulo III

13. Pressione **Next**;
14. Clique em **Next** na tela **Installation Rules**;
15. Mantenha as opções padrões e pressione **Next**;



16. Pressione **Next**;

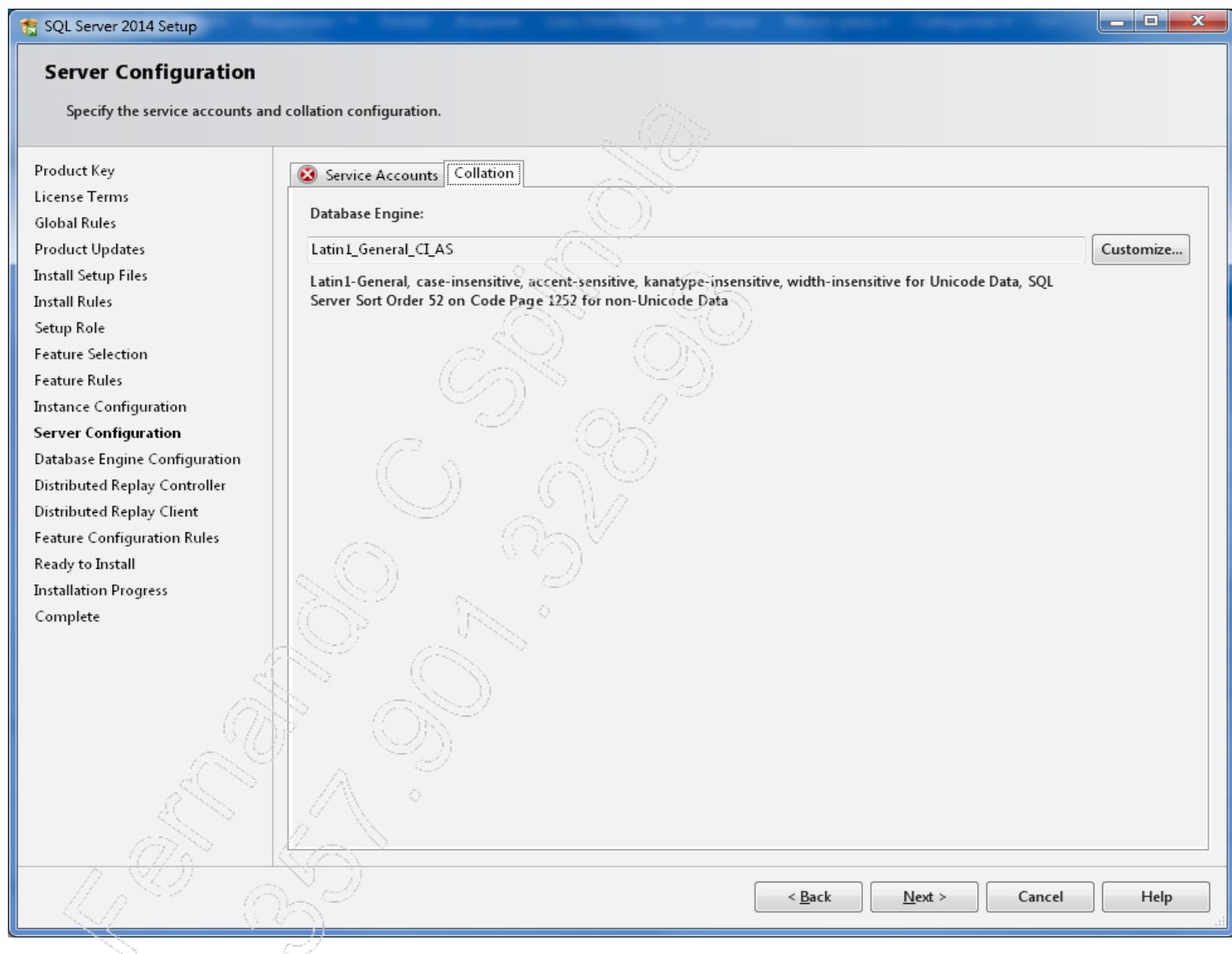
17. Na tela **Server Configuration**, informe os seguintes dados:



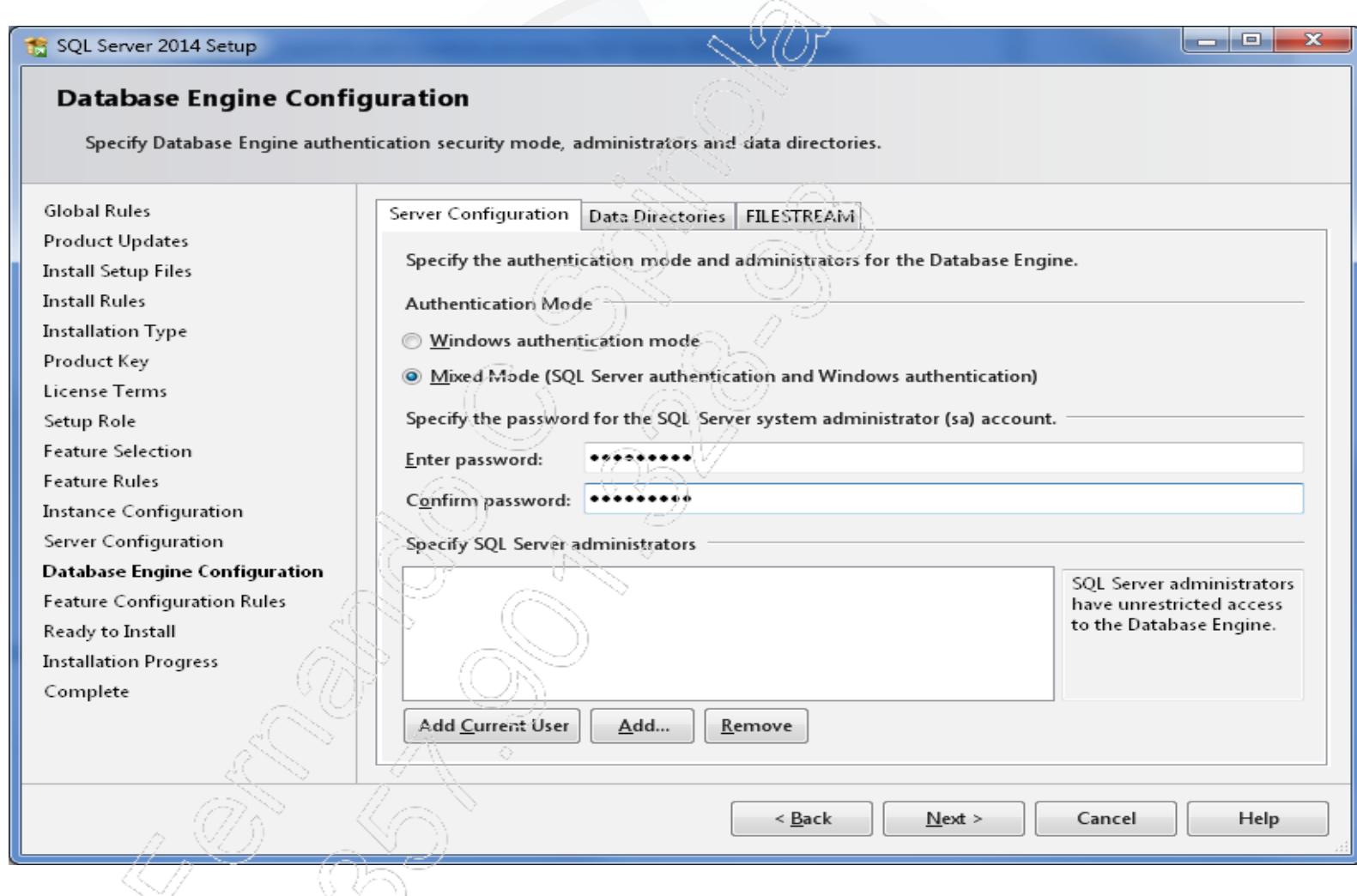
Service	Account Name	Password	Startup Type
SQL Server Agent	Universo\SERVICOSQL	Imp@ct@	Automatic
SQL Server Database Engine	Universo\SERVICOSQL	Imp@ct@	Automatic
SQL Server Browser	Universo\SERVICOSQL	Imp@ct@	Disabled

SQL 2014 - Módulo III

18. Na aba **Collation**, mantenha a opção do Collation do servidor e pressione **Next**:



19. Na tela **Database Engine Configuration**, selecione a opção **Mixed Mode**, informe a senha **Imp@ct@12** (e confirme-a no campo seguinte), pressione o botão **Add Current User**, mantenha as opções das abas **Data Directories** e **FILESTREAM** e pressione **Next**:

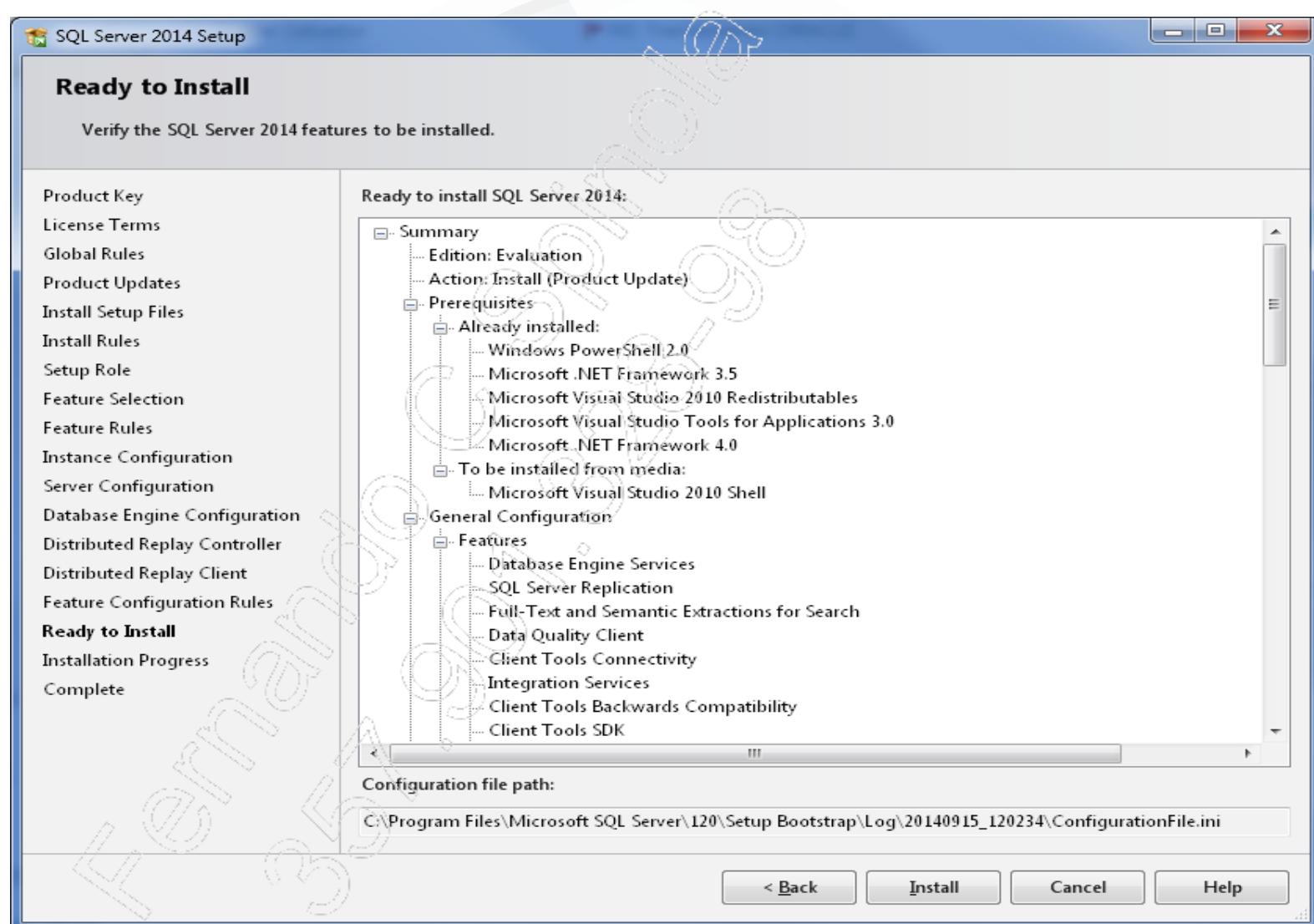


SQL 2014 - Módulo III

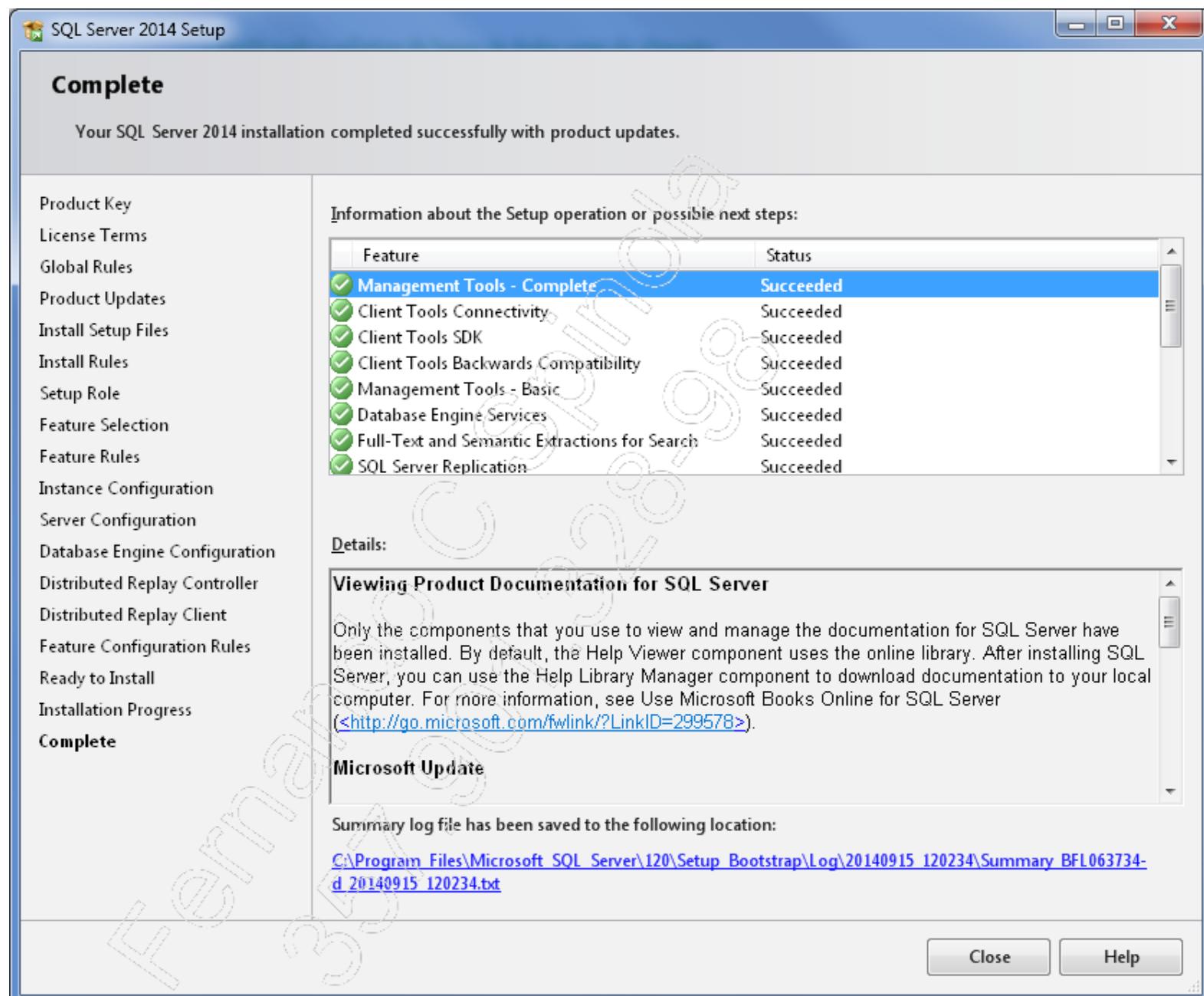
20. Pressione **Next**;

21. Pressione **Next** na tela **Installation Configuration Rules**;

22. Clique em **Install** na tela **Ready to Install**;



23. A instalação do SQL Server está concluída. Clique em **Close** para fechar a tela.

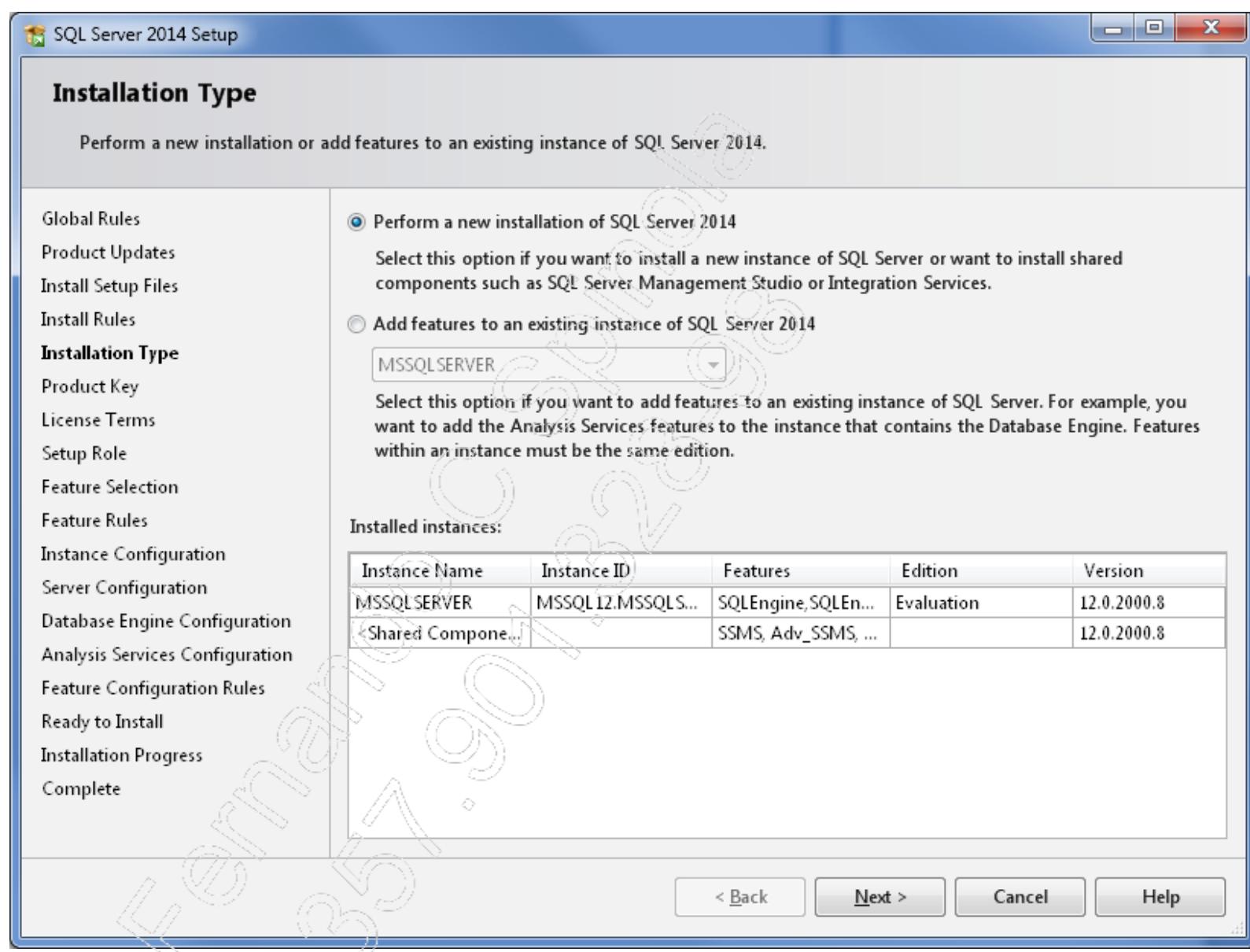


Laboratório 2

A – Instalando uma instância nomeada

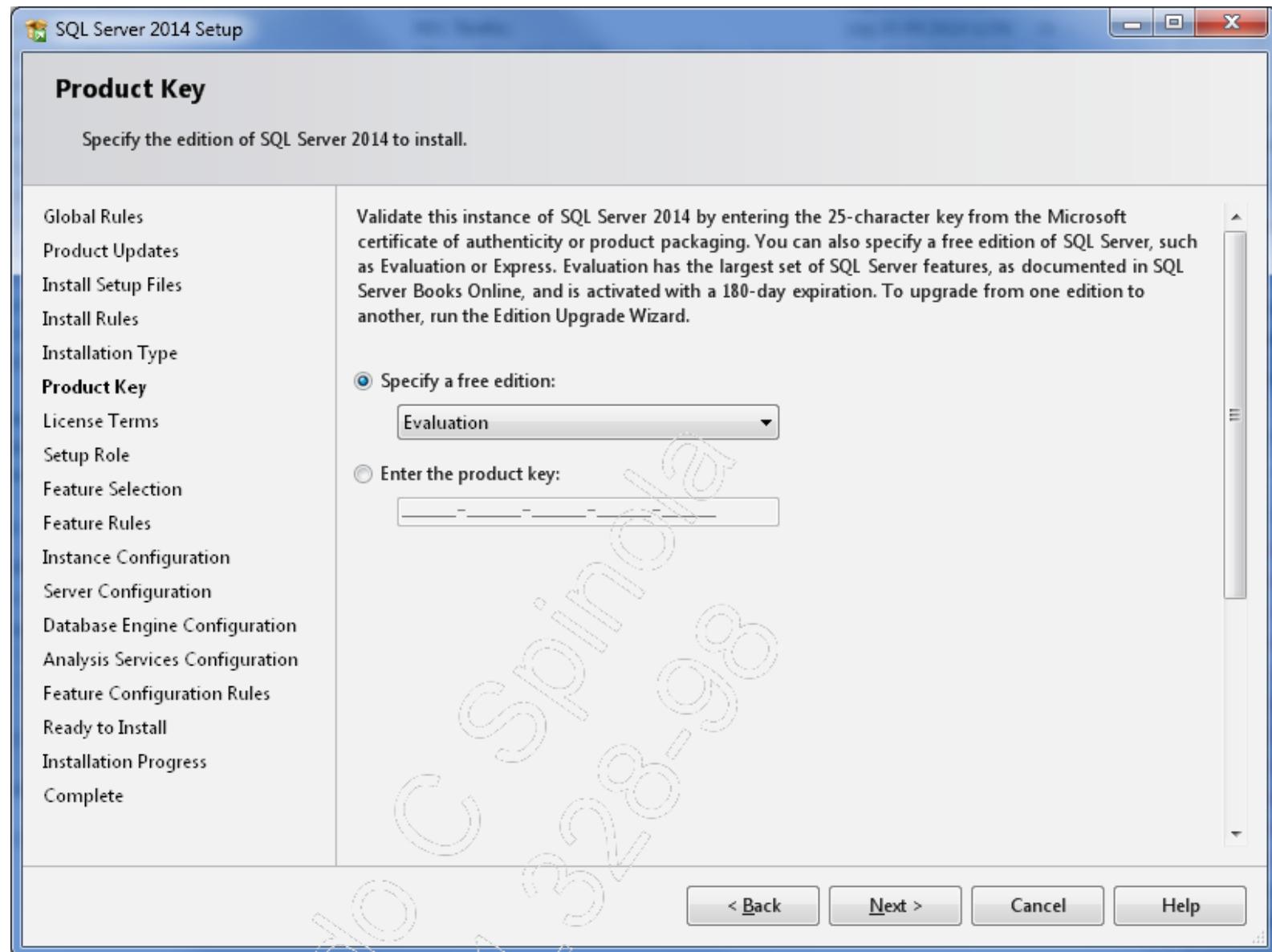
1. No **Windows Explorer**, expanda a pasta **C:\SQL Server 2014 - Módulo III**;
2. Entre na pasta **SQLServer2014**;
3. Efetue um duplo-clique sobre o arquivo **SETUP** e aguarde até que a tela de instalação seja carregada;
4. Clique na opção **Installation** na guia à esquerda;
5. Escolha a opção **New SQL Server stand-alone installation or add features to an existing installation**;
6. Pressione **Ok** na tela **Setup Support Rules**;
7. Especifique a edição utilizada e clique em **Next**;
8. Selecione a opção **Include SQL Server product updates** para incluir as últimas atualizações;
9. Clique em **Next** na tela **Setup Support Rules**;

10. Selecione a opção **Perform a new installation SQL Server 2014** para realizar uma nova instalação e pressione **Next**:



SQL 2014 - Módulo III

11. Especifique a licença Enterprise Evaluation e clique em Next;



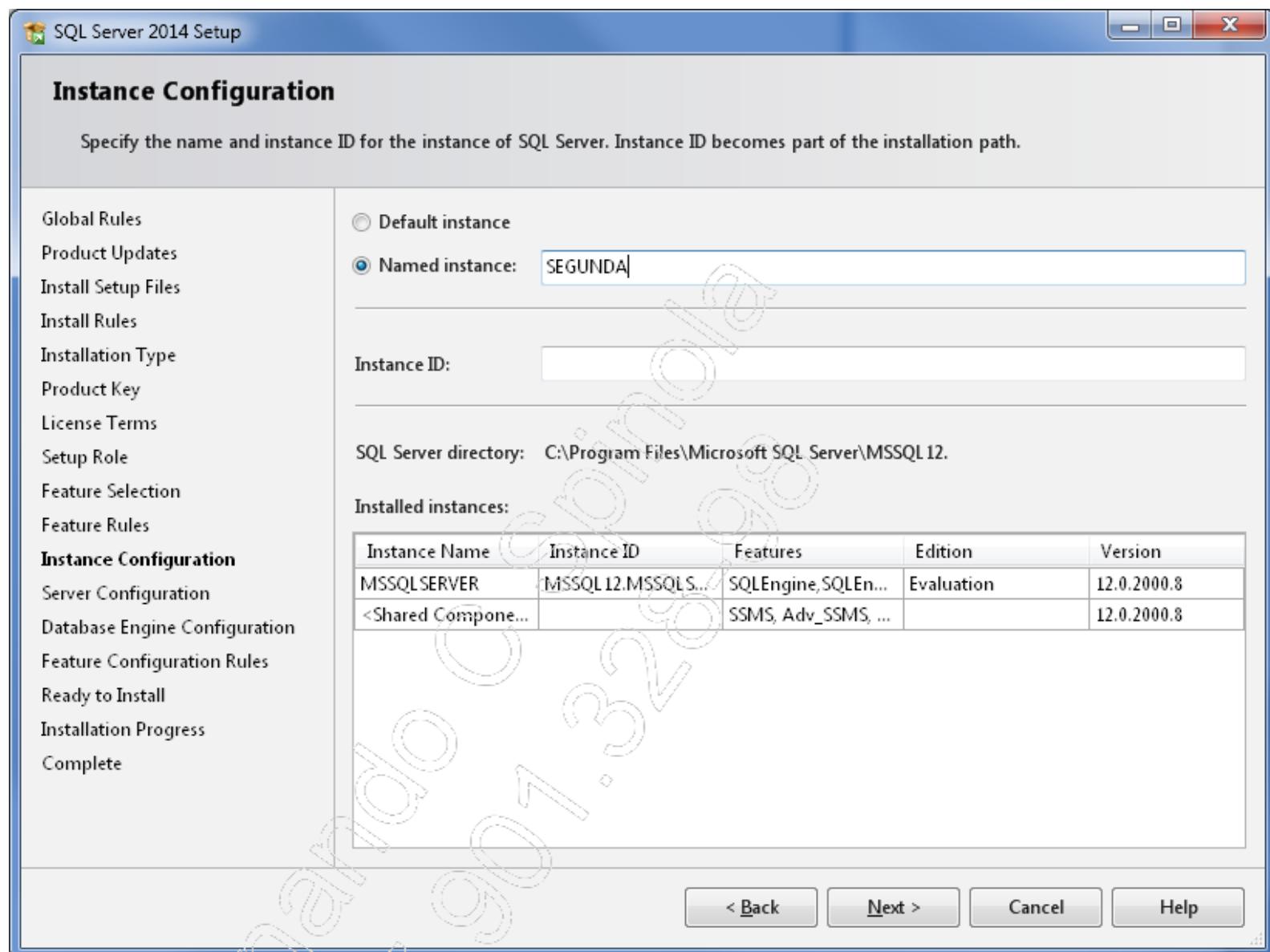
12. Efetue a leitura dos termos de licença, selecione a opção I accept the license terms e clique em Next;

13. Selecione a opção SQL Server Feature Installation para selecionar a opção de instalação e pressione Next;

14. Na tela de Seleção de Opções (Feature Selection), selecione as opções a seguir:

- Instance Features
 - Database Engine Services
 - SQL Server Replication

15. Pressione **Next**;
16. Clique em **Next** na tela **Installation Rules**;
17. Informe o nome da instância nomeada como **SEGUNDA** e pressione **Next**;



18. Verifique os requisitos de disco e clique em **Next**;

SQL 2014 - Módulo III

19. Na tela **Server Configuration**, informe os seguintes dados:

Service	Account Name	Password	Startup Type
SQL Server Agent	Universo\SERVICOSQL	Imp@ct@	Automatic
SQL Server Database Engine	Universo\SERVICOSQL	Imp@ct@	Automatic
SQL Server Browser	Universo\SERVICOSQL	Imp@ct@	Disabled

20. Pressione **Next**;

21. Na tela **Database Engine Configuration**, selecione a opção **Mixed Mode**, informe a senha **Imp@ct@12** (e confirme-a no campo seguinte), pressione o botão **Add Current User**, mantenha as opções das abas **Data Directories** e **FILESTREAM** e pressione **Next**;

22. Pressione **Next** na tela **Error Reporting**;

23. Pressione **Next** na tela **Installation Configuration Rules**;

24. Clique em **Install** na tela **Ready to Install**;

25. A instalação da instância nomeada do SQL Server está concluída. Clique em **Close** para fechar a tela.

Instância e banco de dados 2

- ✓ Instância;
- ✓ Bancos de dados;
- ✓ Banco de dados de sistema;
- ✓ Grupo de arquivos (FileGroup);
- ✓ Criação de um banco de dados;
- ✓ Arquivo de log;
- ✓ Modificação de um banco de dados;
- ✓ Página de dados e extensão;
- ✓ Configurações;
- ✓ Gerenciamento de memória.

2.1. Instância

Instância é um recurso computacional presente em grande parte dos bancos de dados relacionais. Por permitir o armazenamento das informações em memória antes do acesso físico aos dados em um subsistema de disco, as aplicações que a utilizam não precisam acessar os dados diretamente dos arquivos de dados. Esse recurso admite que pelo menos parte de um banco de dados fique em memória.

A instância pode ser configurada pelo administrador e conter mais de um banco de dados. Esses bancos estão ligados à instância e compartilham dos recursos que ela disponibiliza. O encerramento da instância provoca a interrupção no acesso aos bancos de dados nela contidos.

A configuração de uma instância ocorre por meio de parâmetros, os quais permitem definir características como tamanho de alocação de memória para a instância, restrição para acesso a ela, tipos de autenticação de logon suportados pela instância, entre outros.

A seguir, o quadro indica quanto reservar para o sistema operacional e quanto reservar, no máximo, para a instância SQL Server, de acordo com a quantidade de memória do servidor. Caso exista mais do que uma instância, esse valor deve ser a soma delas.

MEMÓRIA TOTAL SISTEMA (GB)	MEMÓRIA RESERVADA SO (GB)	MEMÓRIA TOT SQL SERVER (GB)
16	4	12
32	6	26
64	8	56
128	16	112
256	16	240

2.2. Banco de dados

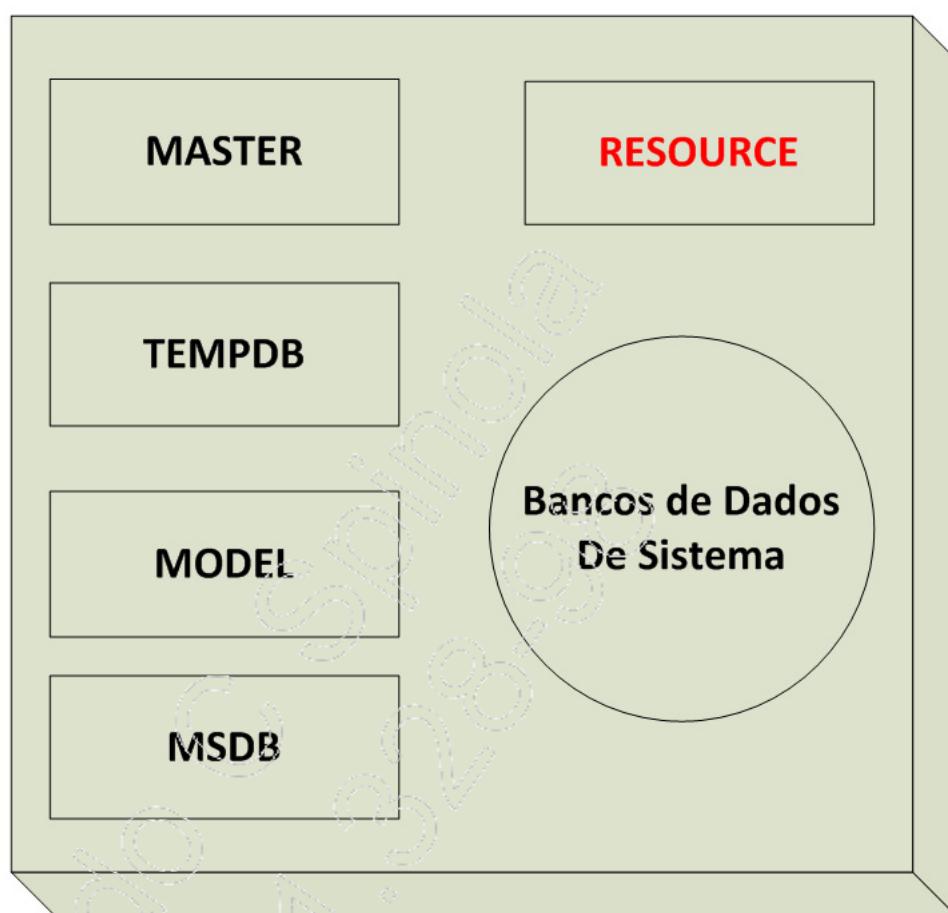
O SQL Server armazena as informações de forma permanente na estrutura denominada banco de dados (database), que é composta de duas partes: os arquivos de dados (datafiles) e os arquivos de log (TLOG).

Os arquivos de dados armazenam os dados assim como os índices criados no banco de dados. Cada arquivo de dados está associado a uma única estrutura denominada grupo de arquivos (FileGroup). Um arquivo associado a um grupo de arquivos não poderá alterar esse grupo após a sua criação.

Os arquivos de log servem para armazenar as transações que ocorrem em um banco de dados SQL Server. Eles são importantes em diversas situações, por exemplo, na recuperação de banco de dados, quando o banco de dados ou a instância é interrompida de forma anormal, na restauração do backup de um banco de dados sem perda de dados, ou ainda na recuperação do banco de dados em um tempo específico.

2.3. Banco de dados de sistema

Toda instância possui quatro bancos de dados (databases), denominados públicos ou de sistema, que são, por natureza, compartilhados entre os bancos de dados criados para armazenamento de dados, chamados de privados. Os bancos de dados públicos são descritos a seguir:



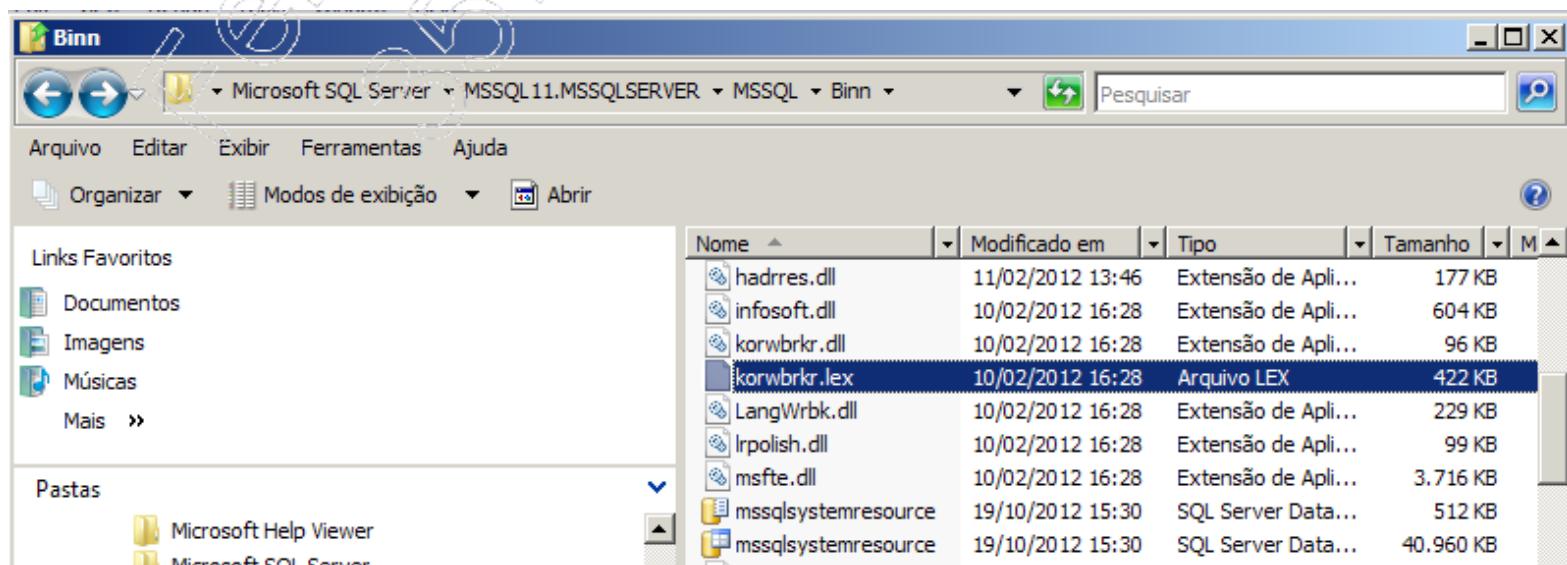
- **Master:** Banco de dados usado para manter os metadados de todos os bancos de dados, incluindo visões internas, tais como `sys.databases`. Objetos como tabelas, índices e stored procedures podem eventualmente ser criados no banco de dados Master, porém, recomenda-se que esses tipos de objeto sejam criados nos bancos de dados privados;
- **Model:** Banco de dados padrão para criação de novos bancos de dados (databases), incluindo nomenclatura, tamanho, estruturas de arquivos e logs. Utilizado para criar banco de dados que não possuem definição de nenhuma estrutura. No caso do comando `CREATE DATABASE IMPACTA`, por exemplo, não definimos tamanho, estruturas, localização dos arquivos, grupo de arquivos (FileGroups) e nenhum outro recurso, pois o **Model** é responsável por isso;

- **Msdb:** Banco de dados de sistema que contém informações sobre SQL Server Agent, assim como as tarefas (jobs) criadas, operadores (operators), alertas (alerts) e todo o histórico de execução das tarefas. É um banco de dados que possui muitas informações importantes para a atividade do SQL Server. Por isso, é importantíssimo manter backups regulares desse banco de dados;
- **TempDB:** Banco de dados de sistema que contém informações temporárias e cujo objetivo é criar objetos de forma temporária. Cada vez que o banco de dados for reiniciado, o TempDB será recriado, vazio e com o tamanho originalmente concebido. Objetos criados neste banco de dados não são registrados em transações no arquivo de log.

A partir da versão 2005, foi adicionado mais um banco de dados de sistema chamado **Resource**, que é apenas para leitura e que armazena todas as estruturas de metadados internas do SQL Server. Isso torna esse banco de dados semelhante ao Master, porém, há diferenças entre eles:

- Resource não pode ser aberto diretamente no SSMS, já o Master deve ser aberto quando a instância é acessada;
- Resource não armazena dados de bancos de dados privados, já o Master armazena essas informações;
- Resource é criado no momento da instalação do banco de dados e não sofre alterações constantes, já o Master sofre alterações constantes.

O database Resource se encontra no diretório **C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\binn**.



2.4. Grupo de arquivos (FileGroup)

Arquivos de dados (datafiles) são os componentes físicos de armazenamento do SQL Server. Todo banco de dados deve ter ao menos um arquivo de dados, e este arquivo pertence a somente um único banco de dados.

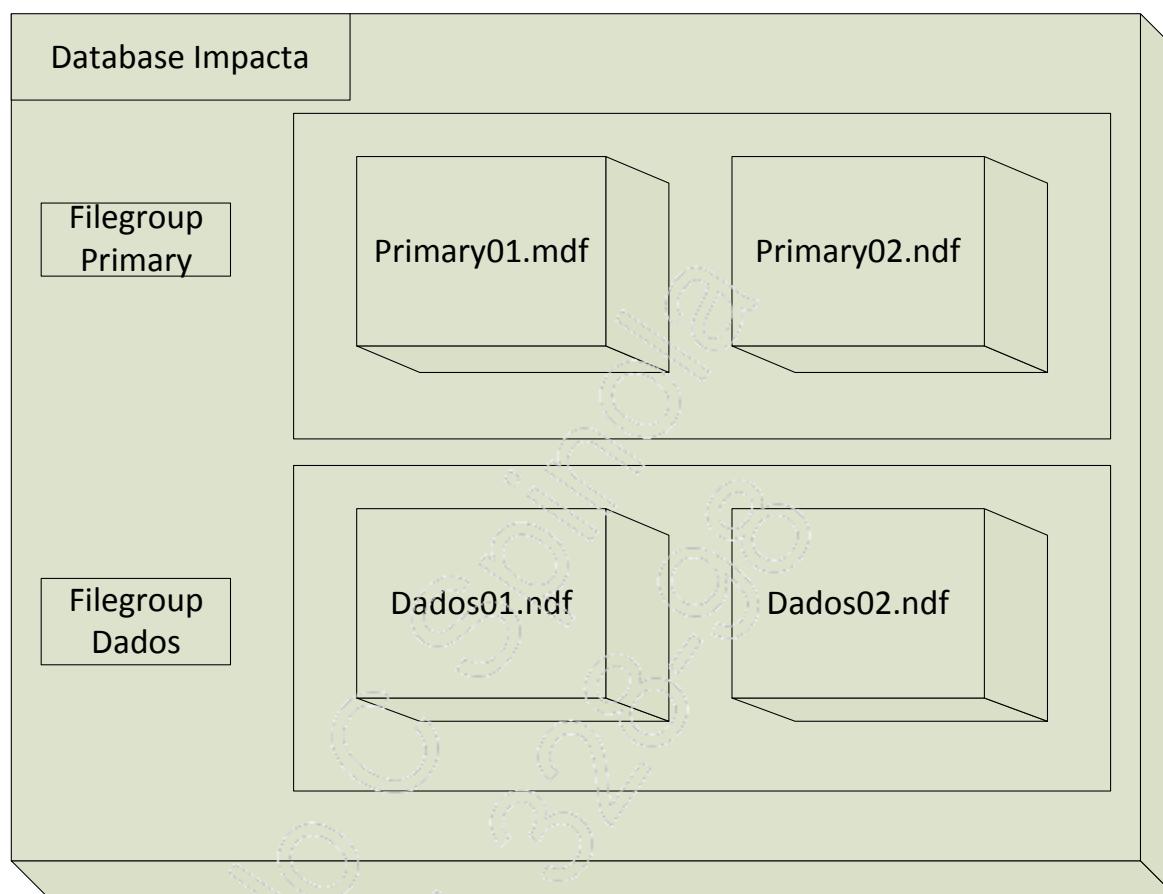
Arquivos de dados estão ligados a uma estrutura lógica de gerenciamento chamada de grupo de arquivos (**FileGroup**). Cada grupo de arquivos pode ter, por sua vez, um ou mais arquivos de dados. O grupo de arquivos serve para organizar os arquivos de dados permitindo que sejam gerenciados de forma única.

Todo banco de dados tem ao menos um grupo de arquivos, que é chamado de **Primary**. É possível criar mais grupos de arquivos e depois associá-los a novos arquivos criados no banco de dados.

Existem várias maneiras de definir a criação de grupos de arquivos. Descrevemos algumas delas a seguir:

- FileGroup **Dados** para armazenamento de tabelas e FileGroup **Índices** para armazenamento de índices;
- FileGroup **Dados_Leitura** para armazenamento de tabelas usadas para leitura, FileGroup **Dados** para armazenamento das demais tabelas e FileGroup **Índices** para armazenamento de índices;

- FileGroup **Sistema_A_Dados** para armazenamento das tabelas referentes ao sistema A e **Sistema_B_Dados** para armazenamento das tabelas referentes ao sistema B.



Na figura anterior, notamos dois grupos de arquivos contendo dois arquivos de dados cada um.

Para obtermos informações sobre os FileGroups:

```
USE nome_database  
SELECT * FROM SYS.SYSFILEGROUPS
```

Para obtermos informações sobre os datafiles:

```
SELECT * FROM SYS.SYSFILES
```

SQL 2014 - Módulo III

Para obtermos informações sobre os FileGroups e seus datafiles:

```
SELECT A.NAME,A.FILE_NAME,A.FILE_ID, B.GROUPNAME  
FROM SYS.SYSFILES A , SYS.SYSFILEGROUPS B  
WHERE A.GROUPID = B.GROUPID
```

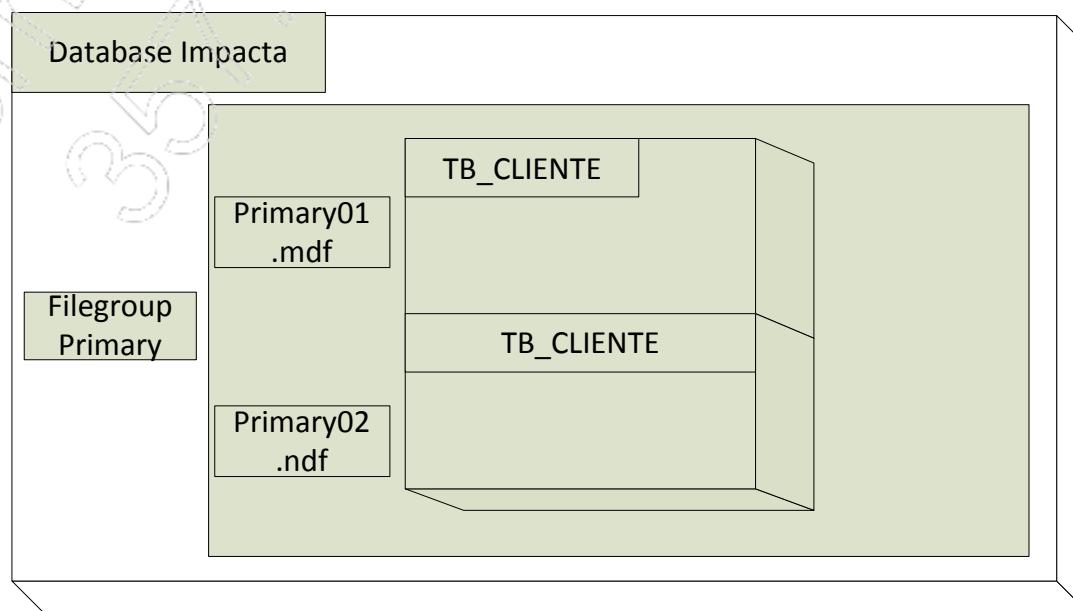
Para criarmos um novo grupo de arquivos em um banco de dados existente:

```
ALTER DATABASE [teste] ADD FILEGROUP [DADOS]  
GO  
ALTER DATABASE [teste] ADD FILEGROUP [DADOS_LEITURA]  
GO  
ALTER DATABASE [teste] ADD FILEGROUP [INDICES]  
GO
```

Para adicionarmos um arquivo de dados em um grupo de arquivos criado:

```
USE [master]  
GO  
ALTER DATABASE [teste]  
ADD FILE ( NAME = N'teste02',  
FILENAME =  
N'C:\Program Files\Microsoft SQL  
Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\teste02.ndf' ,  
SIZE = 4096KB , FILEGROWTH = 1024KB ) TO FILEGROUP [DADOS]  
GO
```

Caso não usemos essa opção para criarmos uma tabela associada ao grupo de arquivos, as tabelas serão criadas no grupo de arquivos **Primary**.



Na figura anterior, temos uma tabela chamada **TB_CLIENTE** que foi criada no FileGroup **Primary** e ocupa parte dos dois arquivos de dados.

Cada tabela criada no banco de dados possui um mapa de arquivo próprio, que tem a função de indicar em quais arquivos físicos a tabela está armazenada. Esse mapa de arquivo tem como finalidade o preenchimento proporcional dos arquivos de um único grupo de arquivo.

- **Resumo das características dos grupos de arquivos:**
 - Não é possível que um arquivo pertença a mais de um grupo de arquivos simultaneamente;
 - Grupos de arquivos não incluem arquivos de log em função do tipo de gerenciamento usado nesses arquivos;
 - Permite alocar objetos específicos;
 - Todo banco de dados possui um grupo de arquivos chamado **Primary**;
 - Pode-se usar o grupo de arquivos para transformar determinados objetos em objetos para simples leitura. Esses objetos não podem ser atualizados;
 - Podemos realizar backups a partir de grupos de arquivos;
 - Apenas tabelas e índices podem ser indicados para um grupo de arquivos desejado.

2.4.1. Tipos de grupos de arquivos

Existem dois tipos de grupos de arquivos: FileGroup padrão e os definidos por usuário:

- **FileGroup Padrão:** Inclui o arquivo de dados primário (**Primary Datafile**) e todos os arquivos de dados não atribuídos a nenhum outro grupo de arquivos. Nesse grupo de arquivos são armazenadas as tabelas de sistema (dicionário de dados);
- **FileGroup definido pelo usuário:** São grupos criados por meio da instrução **FileGroup** em comandos **Create Database** e **Alter Database**.

2.4.2. Recomendações para a divisão de arquivos

É recomendável que as tabelas de sistema sejam armazenadas nos arquivos de dados primários (.mdf). Já no caso de objetos, como tabelas e índices, é indicado armazená-los em arquivos secundários e, neste caso, em grupos de arquivos diferentes, pois, se não fizermos isso, o SQL Server tende a ocupar os arquivos de dados de maneira uniforme.

Sempre que possível, organize dados e índices em grupos de arquivos separados, preferencialmente em discos diferentes, possibilitando um acesso paralelo aos dados.

Tabelas constantemente utilizadas em operações de junção podem ser separadas em grupos de arquivos diferentes, melhorando o acesso físico aos dados.

2.5. Criando um banco de dados

A seguir, temos a sintaxe para a criação de um banco de dados usando SQL Server 2014:

```
CREATE DATABASE database_name
[ CONTAINMENT = { NONE | PARTIAL } ]
[ ON
    [ PRIMARY ] <filespec> [ ,...n ]
    [ , <FileGroup> [ ,...n ] ]
    [ LOG ON <filespec> [ ,...n ] ]
]
[ COLLATE collation_name ]
[ WITH <option> [,...n] ]
[;]

<option> ::==
{
    FILESTREAM ( <filestream_option> [,...n] )
    | DEFAULT_FULLTEXT_LANGUAGE = { lcid | language_name | language_alias }
    | DEFAULT_LANGUAGE = { lcid | language_name | language_alias }
}
    | NESTED_TRIGGERS = { OFF | ON }
    | TRANSFORM_NOISE_WORDS = { OFF | ON }
    | TWO_DIGIT_YEAR_CUTOFF = <two_digit_year_cutoff>
    | DB_CHAINING { OFF | ON }
    | TRUSTWORTHY { OFF | ON }
}

<filestream_option> ::=
{
    NON_TRANSACTED_ACCESS = { OFF | READ_ONLY | FULL }
    | DIRECTORY_NAME = 'directory_name'
}
```

Para realizar um attach com um banco de dados existente

```
CREATE DATABASE database_name
ON <filespec> [ ,...n ]
FOR { { ATTACH [ WITH <attach_database_option> [ , ...n ] ] }
    | ATTACH_REBUILD_LOG }
[;]
```

SQL 2014 - Módulo III

```
<filespec> ::=  
{  
(  
    NAME = logical_file_name ,  
    FILENAME = { `os_file_name` | `filestream_path` }  
    [ , SIZE = size [ KB | MB | GB | TB ] ]  
    [ , MAXSIZE = { max_size [ KB | MB | GB | TB ] | UNLIMITED }  
]  
    [ , FILEGROWTH = growth_increment [ KB | MB | GB | TB | % ] ]  
)  
}  
  
<FileGroup> ::=  
{  
FILEGROUP FileGroup_name [ CONTAINS FILESTREAM ] [ DEFAULT ]  
    <filespec> [ ,...n ]  
}  
  
<attach_database_option> ::=  
{  
    <service_broker_option>  
    | RESTRICTED_USER  
    | FILESTREAM ( DIRECTORY_NAME = { `directory_name` | NULL } )  
}  
  
<service_broker_option> ::=  
{  
    ENABLE_BROKER  
    | NEW_BROKER  
    | ERROR_BROKER_CONVERSATIONS  
}  
Criando um database snapshot  
CREATE DATABASE database_snapshot_name  
    ON  
    (  
        NAME = logical_file_name,  
        FILENAME = `os_file_name'  
    ) [ ,...n ]  
    AS SNAPSHOT OF source_database_name  
[ ; ]
```

As cláusulas do comando **CREATE DATABASE** são:

- **database_name**

Representa o nome do banco de dados.

- **ON**

Indica o grupo de arquivos em que os arquivos de dados serão criados.

- **PRIMARY**

Grupo de arquivo padrão. Caso não seja especificado que outro grupo de arquivos seja o padrão, este grupo será o padrão dos grupos de arquivos.

- **LOG ON**

Define os arquivos de log do banco de dados. Caso esta cláusula não seja declarada, um único arquivo de log será criado com o tamanho equivalente a 25% do total de todos os arquivos de dados.

- **NAME**

NAME representa o nome lógico do arquivo, que poderá ser usado como referência pela maioria dos comandos que necessitem utilizar o arquivo do banco de dados. Deve ser um nome único no banco de dados.

- **FILENAME**

Nome físico do arquivo de dados, o que inclui o caminho no sistema operacional.

- **SIZE**

Valor padrão definido em MB. Representa o tamanho do arquivo.

- **MAXSIZE**

Valor para limitar o aumento do tamanho do arquivo. Este valor deve ser superior ao parâmetro **SIZE**. Caso não seja especificado, o arquivo terá crescimento automático sempre que houver a necessidade de mais espaço, podendo alocar até a totalidade do disco.

- **UNLIMITED**

Parâmetro que indica que o arquivo terá crescimento ilimitado até os limites físicos de espaço em disco.

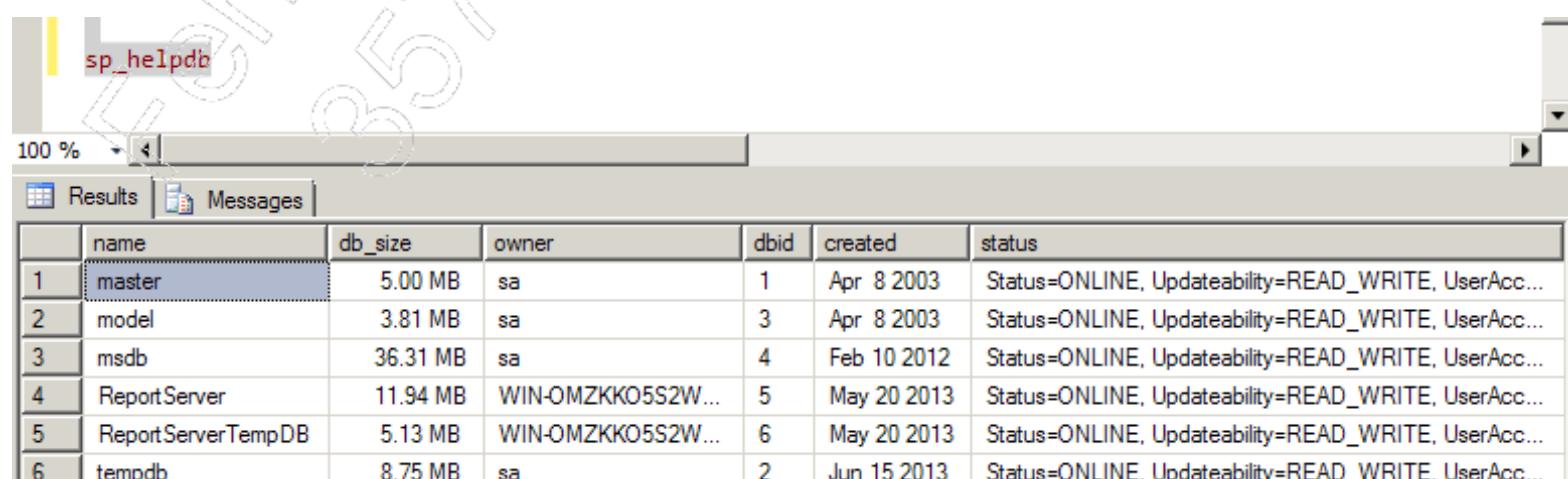
- **FILEGROWTH**

Percentual de aumento ou aumento em valores fixos estabelecidos como padrão em MB, mas que também podem ser indicados em KB, GB e TB. Caso o arquivo seja ilimitado e este parâmetro não seja definido, o valor de incremento deve ser de 1 MB para dados. Já para o arquivo de log, o valor de incremento será de 10% do arquivo ou 64 KB, o que for maior.

2.5.1. Obtendo informações sobre o banco

Existem quatro system stored procedures que retornam informações sobre o banco de dados:

- **sp_helpdb**: informações sobre o database;



	name	db_size	owner	dbid	created	status
1	master	5.00 MB	sa	1	Apr 8 2003	Status=ONLINE, Updateability=READ_WRITE, UserAcc...
2	model	3.81 MB	sa	3	Apr 8 2003	Status=ONLINE, Updateability=READ_WRITE, UserAcc...
3	msdb	36.31 MB	sa	4	Feb 10 2012	Status=ONLINE, Updateability=READ_WRITE, UserAcc...
4	ReportServer	11.94 MB	WIN-OMZKK05S2W...	5	May 20 2013	Status=ONLINE, Updateability=READ_WRITE, UserAcc...
5	ReportServerTempDB	5.13 MB	WIN-OMZKK05S2W...	6	May 20 2013	Status=ONLINE, Updateability=READ_WRITE, UserAcc...
6	tempdb	8.75 MB	sa	2	Jun 15 2013	Status=ONLINE, Updateability=READ_WRITE, UserAcc...

- **sp_spaceused**: Informações de espaço utilizado pelo banco de dados atual;

	database_name	database_size	unallocated space	
1	master	5.00 MB	0.77 MB	
	reserved	data	index_size	unused
1	3304 KB	1320 KB	1528 KB	456 KB

- **sp_helpFileGroup**: Exibe os grupos de arquivos existentes em um banco de dados;

	groupname	groupid	filecount
1	PRIMARY	1	1

- **sp_helpfile**: Exibe dados dos arquivos de dados e log.

	name	fileid	filename	filegroup	size	maxsize	growth	usage
1	master	1	C:\Program Files\Microsoft SQL Server\MSSQL11.MSS...	PRIMARY	4096 KB	Unlimited	10%	data only
2	mastlog	2	C:\Program Files\Microsoft SQL Server\MSSQL11.MSS...	NULL	1024 KB	Unlimited	10%	log only

2.5.2. Executando o CREATE DATABASE

O comando **CREATE DATABASE** pode ser executado de diversas maneiras. Vamos abordar algumas delas a seguir:

- Criando um banco de dados com um arquivo de dados e um log (padrão):

```
CREATE DATABASE PEDIDOS
ON
(
    NAME = 'PEDIDOS_DADOS',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_DADOS01.MDF',
    SIZE = 10MB,
    MAXSIZE = 50MB,
    FILEGROWTH=5MB
)
LOG ON
(
    NAME = 'PEDIDOS_LOG',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_LOG01.LDF',
    SIZE = 5MB,
    MAXSIZE = 30MB,
    FILEGROWTH=2MB
)
```

- Criando um banco de dados com três arquivos de dados, um grupo de arquivos e dois arquivos de log:

```
CREATE DATABASE PEDIDOS
ON
(
    NAME = 'PEDIDOS_DADOS01',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_DADOS01.MDF',
    SIZE = 10MB,
    MAXSIZE = 50MB,
    FILEGROWTH=5MB
),
(
    NAME = 'PEDIDOS_DADOS02',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_DADOS02.NDF',
    SIZE = 10MB,
    MAXSIZE = 50MB,
    FILEGROWTH=5MB
),
(
    NAME = 'PEDIDOS_DADOS03',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_DADOS03.NDF',
    SIZE = 10MB,
    MAXSIZE = 50MB,
    FILEGROWTH=5MB
)
LOG ON
(
    NAME = 'PEDIDOS_LOG01',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_LOG01.LDF',
    SIZE = 5MB,
    MAXSIZE = 30MB,
    FILEGROWTH=2MB
),
(
    NAME = 'PEDIDOS_LOG02',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_LOG02.LDF',
    SIZE = 5MB,
    MAXSIZE = 30MB,
    FILEGROWTH=2MB
)
```

SQL 2014 - Módulo III

- Criando múltiplos grupos de arquivos:

```
CREATE DATABASE PEDIDOS
ON
(
    NAME = 'PEDIDOS_PRIMARIO',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_PRIMARIO.MDF',
    SIZE = 10MB,
    MAXSIZE = 50MB,
    FILEGROWTH=5MB
),
FILEGROUP DADOS
(
    NAME = 'PEDIDOS_DADOS01',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_DADOS01.NDF',
    SIZE = 10MB,
    MAXSIZE = 50MB,
    FILEGROWTH=5MB
),
FILEGROUP INDICES
(
    NAME = 'PEDIDOS_INDICES01',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_INDICES01.NDF',
    SIZE = 10MB,
    MAXSIZE = 50MB,
    FILEGROWTH=5MB
)
LOG ON
(
    NAME = 'PEDIDOS_LOG01',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_LOG01.LDF',
    SIZE = 5MB,
    MAXSIZE = 30MB,
    FILEGROWTH=2MB
)
```

- Criando um banco de dados apenas especificando o arquivo de dados:

```
CREATE DATABASE PEDIDOS
ON
(
    NAME = 'PEDIDOS_PRIMARIO',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_PRIMARIO.MDF',
    SIZE = 10MB,
    MAXSIZE = 50MB,
    FILEGROWTH=5MB
)
```

- Criando um banco de dados usando como base o banco de dados do Model, incluindo tamanho, nome e localização dos arquivos:

```
CREATE DATABASE PEDIDOS
```

- Criando um banco de dados com arquivos de dados ilimitados:

```
CREATE DATABASE PEDIDOS
ON
(    NAME = 'PEDIDOS_PRIMARIO',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_PRIMARIO.MDF')
```

2.6. Arquivo de log

O arquivo de log, ou de transações, é o arquivo que tem a finalidade de registrar todas as ações ocorridas em um banco de dados SQL Server e é utilizado por ele para ações que envolvem a recuperação do banco de dados após uma falha (**Database Recovery**) ou mesmo para desfazer transações que não foram completadas com sucesso.

Este arquivo, em geral, aumenta de acordo com o volume de transações que ocorrem no banco de dados ou através de operações de criação de objetos, por exemplo, **CREATE INDEX**.

Todo banco de dados deve ter ao menos um arquivo de log, mas pode ter mais de um conforme vimos anteriormente. O procedimento de atualização do arquivo de log é o seguinte:

1. A aplicação envia uma transação de dados;
2. A transação é executada pelo SQL Server;
3. Os dados afetados (quando for o caso) são carregados do disco para o database cache. Caso os dados já tenham sido previamente carregados, eles serão lidos diretamente do database cache;

4. Cada comando é registrado no arquivo de log, exceto o comando **SELECT**. Esse comando **SELECT** só será registrado se possuir a cláusula **INTO**;

5. Um processo denominado **CHECKPOINT**, que ocorre de forma assíncrona, fará com que os dados sejam escritos nas tabelas correspondentes.

O mecanismo de log é utilizado sempre que o banco de dados encerra de forma anormal. Dessa forma, ao reiniciar o banco de dados, ele se mostra inconsistente e o processo chamado **RECOVERY** será acionado para realizar a leitura do log. Essa leitura vai efetivar as transações pendentes que receberam o comando **COMMIT** e desfazer as transações que não foram realizadas usando o mecanismo de **ROLLBACK**.

O mecanismo de log pode sofrer atividade em excesso em três circunstâncias:

- **Carregamento de dados em tabelas indexadas**

Atividades realizadas em tabelas indexadas provocam um aumento de atividade de log, pois todas as atividades são registradas nesse log.

- **Transações que executam modificações em excesso**

Comandos **UPDATE** e **DELETE** sem a cláusula **WHERE** e comando **INSERT** podem provocar excessiva atualização de log em função do volume de dados movimentado.

- **Uso dos comandos WRITETEXT ou UPDATETEXT (WITH LOG)**

Por padrão, esses comandos não movimentam log, exceto se usada a cláusula **WITH LOG**.

2.7. Modificando um banco de dados

Para modificarmos as características de um banco de dados, é necessário usarmos o comando **ALTER DATABASE**. A seguir, veremos a sintaxe básica desse comando:

```
ALTER DATABASE database_name
{
    <add_or_modify_files>
    | <add_or_modify_FileGroups>
}
[;]

<add_or_modify_files>::=
{
    ADD FILE <filespec> [ ,...n ]
        [ TO FILEGROUP { filegroup_name } ]
    | ADD LOG FILE <filespec> [ ,...n ]
    | REMOVE FILE logical_file_name
    | MODIFY FILE <filespec>
}

<filespec>::=
(
    NAME = logical_file_name
    [ , NEWNAME = new_logical_name ]
    [ , FILENAME = { 'os_file_name' | 'filestream_path' } ]
    [ , SIZE = size [ KB | MB | GB | TB ] ]
    [ , MAXSIZE = { max_size [ KB | MB | GB | TB ] | UNLIMITED } ]
    [ , FILEGROWTH = growth_increment [ KB | MB | GB | TB | % ] ]
    [ , OFFLINE ]
)

<add_or_modify_filegroups>::=
{
    | ADD FILEGROUP filegroup_name
        [ CONTAINS FILESTREAM ]
    | REMOVE FILEGROUP filegroup_name
    | MODIFY FILEGROUP filegroup_name
        { <filegroup_updatability_option>
        | DEFAULT
        | NAME = new_filegroup_name
        }
    }
<filegroup_updatability_option>::=
{
    { READONLY | READWRITE }
    | { READ_ONLY | READ_WRITE }
}
```

Em que:

- **ADD FILE**: Define qual arquivo será adicionado ao banco de dados;
- **NAME**: Nome lógico do arquivo de dados ou log, este nome deve ser único dentro do banco de dados;
- **FILENAME**: Nome físico do arquivo de dados ou log;
- **SIZE**: Tamanho inicial do arquivo, cujo padrão é MB, mas pode ser representado em KB, GB e TB;
- **MAXSIZE**: Tamanho máximo do arquivo, cujo padrão é MB, mas pode ser representado em KB, MB, GB e TB;
- **UNLIMITED**: Determina que o tamanho do arquivo é ilimitado, podendo aumentar até ocupar o disco integralmente;
- **FILEGROWTH**: Percentual ou acréscimo de crescimento de um arquivo. Pode ser representado em KB, MB, GB e TB ou por um valor percentual;
- **TO FILEGROUP**: Determina em qual grupo de arquivos o arquivo será associado;
- **ADD LOGFILE**: Adiciona um novo arquivo de log ao banco de dados;
- **REMOVE FILE**: Remove tabelas e o arquivo físico de dados;
- **ADD FILEGROUP**: Adiciona um novo grupo de arquivos;
- **REMOVE FILEGROUP**: Remove um grupo de arquivos;
- **MODIFY FILE**: Altera um arquivo existente nos itens tamanho (**SIZE**), nome do arquivo (**FILENAME**), tamanho máximo do arquivo (**MAXSIZE**) e crescimento do arquivo (**FILEGROWTH**).

2.7.1. Exemplos

Vamos conhecer, a seguir, diversas formas de alterar um banco de dados:

- **Adicionando um novo arquivo ao banco de dados no FileGroup Primary**

```
ALTER DATABASE PEDIDOS
ADD FILE
(
    NAME = 'PEDIDOS_PRIMARIO_02',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_PRIMARIO_02.MDF',
    SIZE = 10MB,
    MAXSIZE = 50MB,
    FILEGROWTH=5MB
)
```

- **Adicionando um novo arquivo ao banco de dados no FileGroup Dados**

```
ALTER DATABASE PEDIDOS
ADD FILE
(
    NAME = 'PEDIDOS_DADOS02',
    FILENAME = 'C:\DATABASES\PEDIDOS\PEDIDOS_DADOS02.NDF',
    SIZE = 10MB,
    MAXSIZE = 50MB,
    FILEGROWTH=5MB
) TO FILEGROUP DADOS
```

- **Indicando um novo grupo de arquivos padrão para o banco de dados**

```
ALTER DATABASE PEDIDOS
MODIFY FILEGROUP DADOS DEFAULT
```

- **Removendo um arquivo**

```
ALTER DATABASE PEDIDOS
REMOVE FILE PEDIDOS_DADOS02
```

- **Removendo um grupo de arquivos**

```
ALTER DATABASE PEDIDOS  
REMOVE FILEGROUP INDICES
```



Não é possível remover o grupo de arquivos **PRIMARY**.

- **Aumentando o tamanho de um arquivo**

```
ALTER DATABASE PEDIDOS  
MODIFY FILE  
( NAME = 'PEDIDOS_PRIMARIO_02',  
    SIZE = 20MB  
)
```

- **Diminuindo o tamanho dos arquivos de um banco de dados**

Dispomos de um comando chamado **DBCC SHRINKDATABASE**, que tem como finalidade a redução de todos os arquivos do banco de dados.

```
DBCC SHRINKDATABASE  
( database_name [,target_percent] [, {NOTRUNCATE | TRUNCATEONLY} ]  
)
```

- **NOTRUNCATE**: Os espaços livres nos arquivos de dados não são liberados para o sistema operacional;
- **TRUNCATEONLY**: Os espaços não utilizados são liberados para o sistema operacional. Reduz os arquivos até a última extensão alocada (**extent**). Caso utilizemos esta opção, o valor para **target_percent** será ignorado.

Exemplo: Reduzindo o banco de dados **PEDIDOS** para 25% do tamanho:

```
DBCC SHRINKDATABASE (PEDIDOS, 25)
```

- Reduzindo o tamanho de um arquivo

```
DBCC SHRINKFILE  
( file_name [,target_size] [, { EMPTYFILE | NOTRUNCATE | TRUNCATE-  
ONLY } ]  
)
```

As opções **NOTRUNCATE** e **TRUNCATEONLY** foram descritas anteriormente. A opção **EMPTYFILE** faz a transferência de todas as páginas que estão em um arquivo de um grupo de arquivos (**FileGroup**) para outro arquivo do mesmo grupo e possibilita a exclusão deste arquivo através do comando **ALTER DATABASE**.

Se não for especificado um valor para **NOTRUNCATE**, o sistema operacional não terá o espaço livre disponível no arquivo.

Exemplo: Reduzindo o arquivo **PEDIDOS_PRIMARIO_02** para 3 MB:

```
DBCC SHRINKFILE (PEDIDOS_PRIMARIO_02, 3)
```

- Alterando o nome de um banco de dados

Exemplo: Alterando o nome do banco de dados **PEDIDOS** para **PEDIDOS_IMPACTA**:

```
ALTER DATABASE PEDIDOS  
MODIFY NAME = PEDIDOS_IMPACTA
```

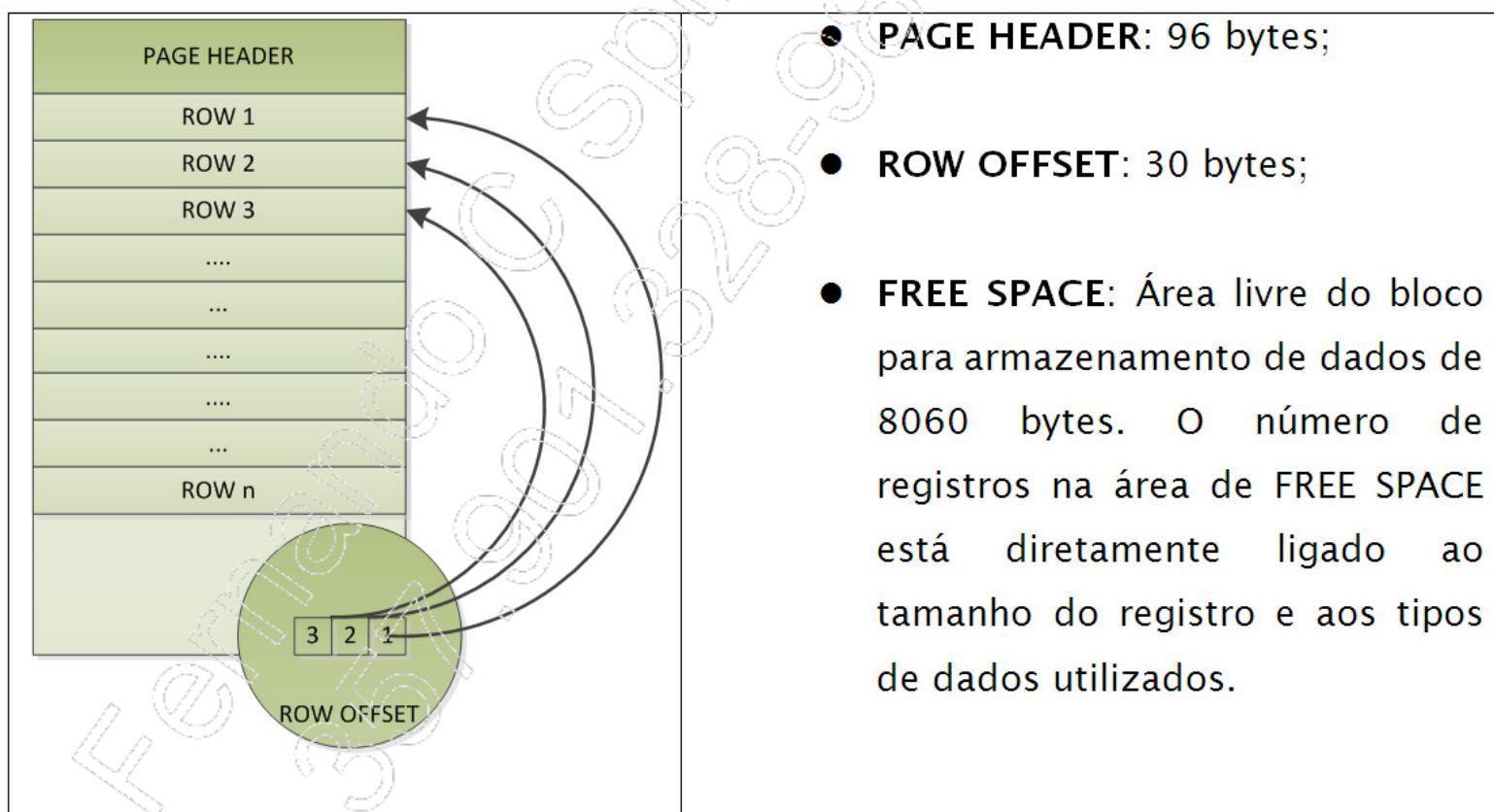
Ou:

```
EXEC SP_RENAMEDB 'PEDIDOS', 'PEDIDOS_IMPACTA'
```

2.8. Página de dados e extensão

A página é a estrutura para armazenamento de dados, e corresponde a 8 KB. Ela é composta pelo cabeçalho (PAGE HEADER), que utiliza 96 bytes de espaço, e por uma área útil para armazenamento de 8060 bytes (excluindo o PAGE HEADER).

A seguir, veremos com mais detalhes a estrutura que você já conhece de uma página de dados no SQL Server:



Existem vários tipos de páginas no SQL Server, os quais serão descritos a seguir:

- **Página de dados:** Página que contém dados (tabelas), exceto colunas dos tipos de dados **text**, **ntext**, **image**, **nvarchar(max)**, **varchar(max)**, **varbinary(max)** e **xml**;
- **Página de índice:** Informações relativas aos índices;
- **Página de texto/imagem:** Para tabelas que contenham tipos de dados **text**, **ntext**, **image**, **nvarchar(max)**, **varchar(max)**, **varbinary(max)** e **xml**. Também pode conter os tipos de dados **varchar**, **nvarchar**, **varbinary** e **sql_variant**, quando estes forem superiores a 8060 bytes;
- **Mapa de alocação Global (GLOBAL ALLOCATION MAP – GAM) e Mapa de Alocação Global Compartilhada (SHARED GLOBAL ALLOCATION MAP – SGAM):** Armazena informações sobre alocações das extensões. Uma página SGAM pode armazenar informações de 64000 extensões (EXTENTS), o que equivale a 4 MB, logo, a cada 4 MB haverá uma página SGAM. O mesmo ocorrerá com as páginas do tipo GAM. Resumindo: as páginas GAM guardam informações de extensões (EXTENTS) uniformes e as SGAM, de extensões (EXTENTS) mistas;
- **Espaço livre em página (FREE PAGE SPACE):** Espaço livre nas páginas;
- **Mapa de alocação de índice (INDEX ALLOCATION MAP – IAM):** Contém informações sobre a alocação de extensões de dados e índices;
- **Bulk Changed copy:** Informações sobre extensões modificadas pelas operações em massa desde a última instrução BACKUP LOG por unidade de alocação;
- **Mapa de alterações diferenciais (Differential Changed Map):** Informações sobre extensões modificadas desde a última instrução BACKUP DATABASE por unidade de alocação.

SQL 2014 - Módulo III

Extensão (EXTENT) é o primeiro tipo de alocação de dados no SQL Server, correspondendo a 8 páginas de dados (64 KB). Isso significa que quando criamos uma tabela ou um índice, eles precisam ser alocados nos arquivos de dados (DATAFILES) e isso ocorre através da alocação de extensão de dados. Há duas formas de extensão (EXTENT):

- **Extensão uniforme:** Alocação de oito páginas de dados para um único objeto, o que equivale a 64 KB alocados dentro do arquivo de dados. Uma extensão está alocada a somente um arquivo de dados, não podendo ser dividida entre dois arquivos de dados (DATAFILES) ao mesmo tempo. Este tipo de extensão é particularmente útil para objetos que alocam grandes quantidades de páginas;
- **Extensão mista:** Também é composta de oito páginas, porém até oito objetos diferentes (tabelas e índices) podem compartilhá-las ao mesmo tempo. Este tipo de alocação é bastante útil quando os objetos e índices possuem um tamanho pequeno.

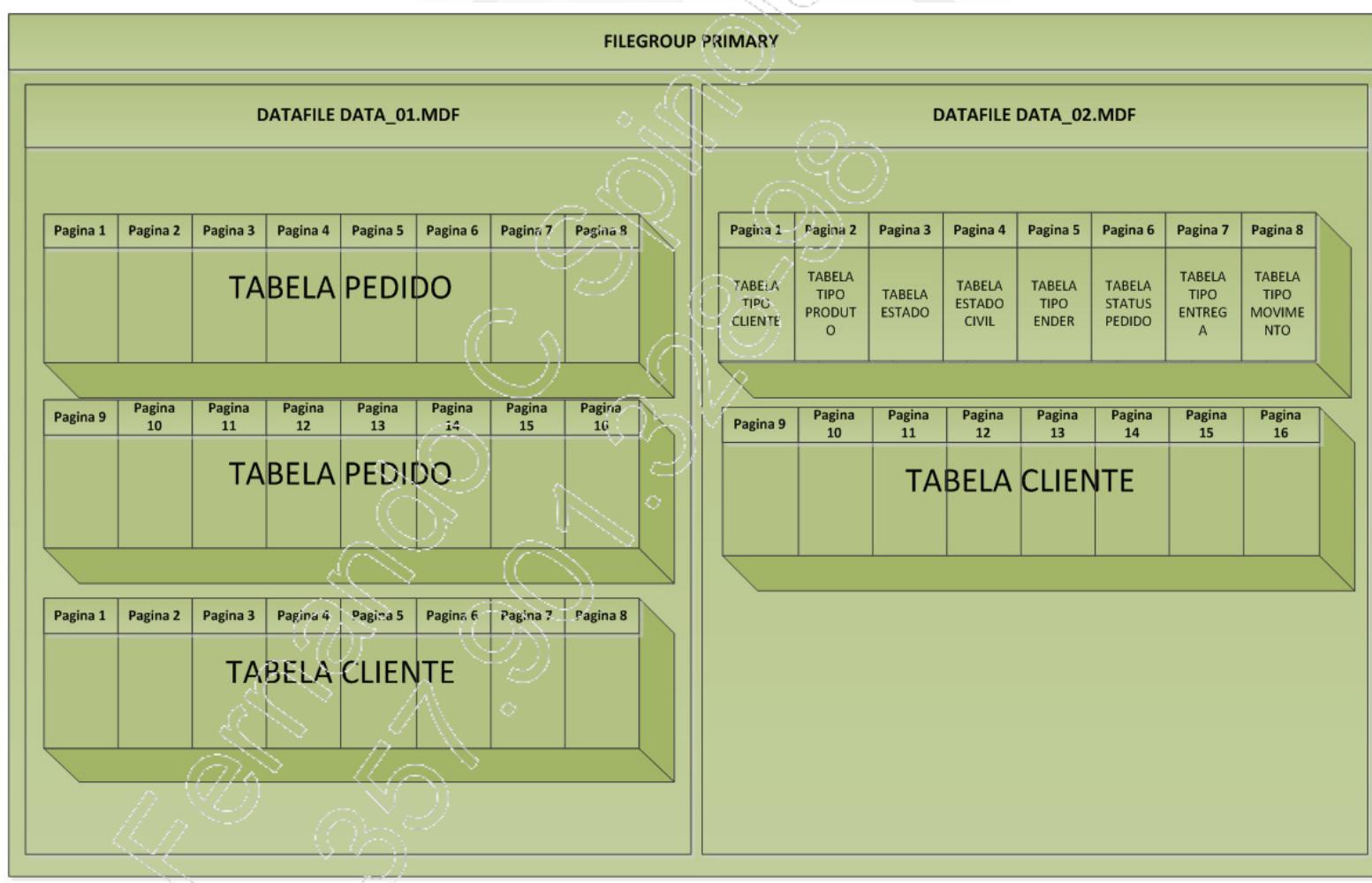
Para demonstrar, a imagem a seguir ilustra a disposição da alocação dos extents e das páginas:



Um objeto que aloca extensão (EXTENT) uniforme alocará apenas extensões desse tipo. O mesmo ocorre se o objeto alocar extensão mista.

A alocação de tabelas ou índices geralmente é feita em extensões mistas, pois elas ocupam menos espaço que as extensões uniformes. Ao crescer e atingir oito páginas mistas, a alocação é alterada para extensão uniforme. O mesmo vale para os índices, porém, se os índices forem criados ou refeitos e caso a tabela possua uma quantidade de registros que possa ocupar oito ou mais páginas, a alocação do índice será integralmente coberta por extensões uniformes.

A imagem a seguir ilustra a hierarquia de armazenamento do SQL Server:



Um banco de dados pode ter um ou mais grupos de arquivos (FILEGROUPS) e cada um desses grupos pode ter um ou mais arquivos de dados (DATAFILES). Esses arquivos podem ter uma ou mais extensões (EXTENTS) e cada extensão possui oito páginas.

SQL 2014 - Módulo III

As tabelas e índices são criados em grupos de arquivos (FILEGROUPS). À medida que o banco de dados cresce, podemos adicionar mais arquivos de dados a um grupo de arquivos e, assim, os objetos poderão aumentar em quantidade e tamanho mesmo que em arquivos de dados diferentes (por exemplo, TABELA CLIENTE).

A seguir, vamos demonstrar a alocação dos dados nas páginas de dados:

- Realizando a inserção na tabela:

```
use pedidos;
insert into marca values (1,'marca1');
insert into marca values (2,'marca2');
insert into marca values (3,'marca3');
insert into marca values (4,'marca4');
insert into marca values (5,'marca5');
insert into marca values (6,'marca6');
insert into marca values (7,'marca7');
insert into marca values (8,'marca8');
insert into marca values (9,'marca9');
insert into marca values (10,'marca10');
```

- Buscando as páginas alocadas para a tabela:

```
DBCC IND('pedidos','marca',-1)
```

- Sintaxe:

```
DBCC IND ( { 'dbname' | dbid }, { 'objname' | objid }, {
nonclustered indid | 1 | 0 | -1 | -2 });
nonclustered indid = ID de índice não clusterizado
/*
1 = ID de índice clusterizado
0 = Exibe informações de registros dentro da página e de páginas
IAM
-1 = Exibe informações de todas as páginas de índices e de
páginas LOB e páginas de estouro (OVERFLOW PAGES)
-2 = Exibe informação de todas as páginas IAM
*/
```

O resultado mostra as páginas alocadas para a tabela e o índice:

	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel
1	1	306	NULL	NULL	469576711	1	1	72057594039500800	In-row data	10	NULL
2	1	305	1	306	469576711	1	1	72057594039500800	In-row data	1	0

- Encontrando informações das páginas localizadas no passo anterior:

```
DBCC TRACEON(3604)
GO
DBCC page('pedidos', 1, 305, 3)
```

- Sintaxe:

```
dbcc page ( { 'dbname' | dbid}, filenum, pagenum [, 
printopt={0|1|2|3} ]);Printopt:
/*
0 - Apenas imprime o Page Header
1 - page header mais dump do array de slot
2 - page header mais conteúdo hexadecimal da página
3 - page header mais dados de cada linha da página
*/
```

SQL 2014 - Módulo III

O resultado (resumido) mostra a alocação dentro da página 305:

```
DBCC execution completed. If DBCC printed error messages, contact your
system administrator.
```

```
PAGE: (1:305)
```

```
BUFFER:
```

```
BUF @0x00000000479E280
```

```
bpage = 0x0000000176070000      bhash = 0x0000000000000000
bpageno = (1:305)                bpreferences = 0
bdbid = 8                         bUse1 = 47844
bcputicks = 0                      bnnext = 0x0000000000000000
bsampleCount = 0
bstat = 0x10b
blog = 0x1c9a
```

```
PAGE HEADER:
```

```
Page @0x0000000176070000
```

```
m_pageId = (1:305)               m_headerVersion = 1           m_
type = 1                           m_level = 0                 m_
m_typeFlagBits = 0x0              m_indexId (AllocUnitId.idInd) =
flagBits = 0x8000
m_objId (AllocUnitId.idObj) = 91   m_indexId (AllocUnitId.idInd) =
256
Metadata: AllocUnitId = 72057594043891712
Metadata: PartitionId = 72057594039500800
Metadata: IndexId = 1
Metadata: ObjectId = 469576711     m_prevPage = (0:0)          m_
nextPage = (0:0)                  m_slotCnt = 10            m_
pminlen = 9                       m_reservedCnt = 0        m_
freeCnt = 7853
m_freeData = 319
lsn = (31:961:2)
m_xactReserved = 0
ghostRecCnt = 0
m_tornBits = -729153226          m_xdesId = (0:0)          m_
                                         DB Frag ID = 1
```

```
Allocation Status
```

```
GAM (1:2) = ALLOCATED           SGAM (1:3) = ALLOCATED
PFS (1:1) = 0x60 MIXED_EXT ALLOCATED 0_PCT_FULL
DIFF (1:6) = CHANGED
ML (1:7) = NOT MIN_LOGGED
```

```
Slot 0 Offset 0x60 Length 24

Record Type = PRIMARY_RECORD          Record Attributes = NULL_BITMAP
VARIABLE_COLUMNS
Record Size = 24
Memory Dump @0x000000003D4AA060

0000000000000000: 30000900 01010000 00020000 01001800 42524153 0.
.....BRAS
0000000000000014: 54454d50                      TEMP

Slot 0 Column 1 Offset 0x4 Length 5 Length (physical) 5
COD_MARCA = 1.

Slot 0 Column 2 Offset 0x10 Length 8 Length (physical) 8
DES_MARCA = BRASTEMP

Slot 0 Offset 0x0 Length 0 Length (physical) 0

KeyHashValue = (06e6c85f0c03)
Slot 1 Offset 0x78 Length 22

Record Type = PRIMARY_RECORD          Record Attributes = NULL_BITMAP
VARIABLE_COLUMNS
Record Size = 22
Memory Dump @0x000000003D4AA078

0000000000000000: 30000900 01020000 00020000 01001600 6d617263 0.
.....marc
0000000000000014: 6132                  a2

Slot 1 Column 1 Offset 0x4 Length 5 Length (physical) 5
COD_MARCA = 2.

Slot 1 Column 2 Offset 0x10 Length 6 Length (physical) 6
DES_MARCA = marca2

Slot 1 Offset 0x0 Length 0 Length (physical) 0

.....
DBCC execution completed. If DBCC printed error messages, contact your
system administrator.
```

Em qualquer arquivo de dados (DATAFILE), existe uma ordem para alocação das páginas de controle. Vejamos a ordem dessas páginas:

- Página 0 – Cabeçalho do arquivo (HEADER PAGE);
- Página 1 – Espaço livre de página (PAGE FREE SPACE);
- Página 2 – Mapa de alocação global (GLOBAL ALLOCATION MAP – GAM);
- Página 3 – Mapa de alocação global compartilhada (SHARED GLOBAL ALLOCATION MAP – SGAM).

2.9. Configurações

Vejamos, a seguir, as opções para configuração de um banco de dados SQL Server 2014:

- **ANSI_NULL_DEFAULT** (Default **OFF**): Quando criarmos tabelas e não especificarmos explicitamente se a coluna é **NOT NULL**, a coluna será considerada **NOT NULL** caso o valor desse parâmetro seja **OFF**. Agora, se o valor for **ON**, a coluna será nula;
- **ANSI_NULLS** (Default **OFF**): Qualquer comparação feita usando valor nulo será avaliado como **UNKNOWN** caso a opção **ON** seja definida. Se a opção definida for **OFF**, as comparações entre valores nulos que não forem **UNICODE** terão como resultado **TRUE**;
- **ANSI_PADDING**: Quando definimos o valor **ON**, não há organização de valores em branco para colunas **varchar** ou **nvarchar**. Para colunas **char**, **nchar** e **binary**, valores nulos são usados para preenchimento da coluna. Quando o valor é marcado como **OFF**, as trilhas em branco são organizadas em valores para caracteres inseridos em colunas **varchar** ou **nvarchar** e trilhas de zeros em valores binários, inseridos em colunas **varbinary**;

- **ANSI_WARNINGS** (Default **OFF**): Caso o valor seja **ON**, quando há ocorrência de valores nulos em funções de agrupamento (**SUM**, **MIN**, **MAX**, **AVG** e **COUNT**), ou divisão por zero, há relato de erro ou aviso de erro. Caso seja indicado o valor **OFF**, nenhum dos casos indicados anteriormente ocorre e o resultado será sempre nulo;
- **ARITHABORT**: Caso a opção seja **ON**, em caso de divisão por zero, ocorrerá a finalização do comando durante a execução. Caso a opção seja **OFF**, surgirá uma mensagem de erro, mas a transação ou consulta continuará normalmente;
- **CONCAT_NULL_YIELDS_NULL** (Default **OFF**): Caso a opção seja **ON**, em uma concatenação, se um dos valores for nulo, o resultado será nulo. Caso a opção seja **OFF**, o valor nulo será concatenado normalmente;
- **QUOTED_IDENTIFIER** (Default **OFF**): A opção **ON** permite o uso de aspas duplas ("") para envolver identificadores delimitados. Caso seja **OFF**, não será possível o uso de aspas duplas, além disso, todos os identificadores devem obedecer às regras de **transact-sql**;
- **NUMERIC_ROUNDABORT** (Default **OFF**): Caso utilizemos a opção **ON**, em caso de perda de precisão em uma expressão, um erro será gerado. Caso o valor seja **OFF**, em caso de perda de precisão, o valor será arredondado;
- **RECURSIVE_TRIGGERS** (Default **OFF**): O valor **ON** permite recursividade no disparo de gatilhos, já o valor **OFF** não permite a recursividade;
- **AUTO_CLOSE** (Default **OFF**): Caso o valor seja **ON**, após o último usuário desconectar do banco, este último será fechado (**CLEAN SHUTDOWN**). Se outro usuário tentar utilizar o banco de dados, ele será automaticamente aberto. Caso o valor seja **OFF**, o banco de dados continuará aberto mesmo após a saída do último usuário;
- **AUTO_CREATE_STATISTICS** (Default **ON**): Caso a opção seja **ON**, as estatísticas para o banco de dados serão geradas para as colunas usadas em predicado (**WHERE**). Caso seja **OFF**, a coleta de estatísticas deverá ser manual;

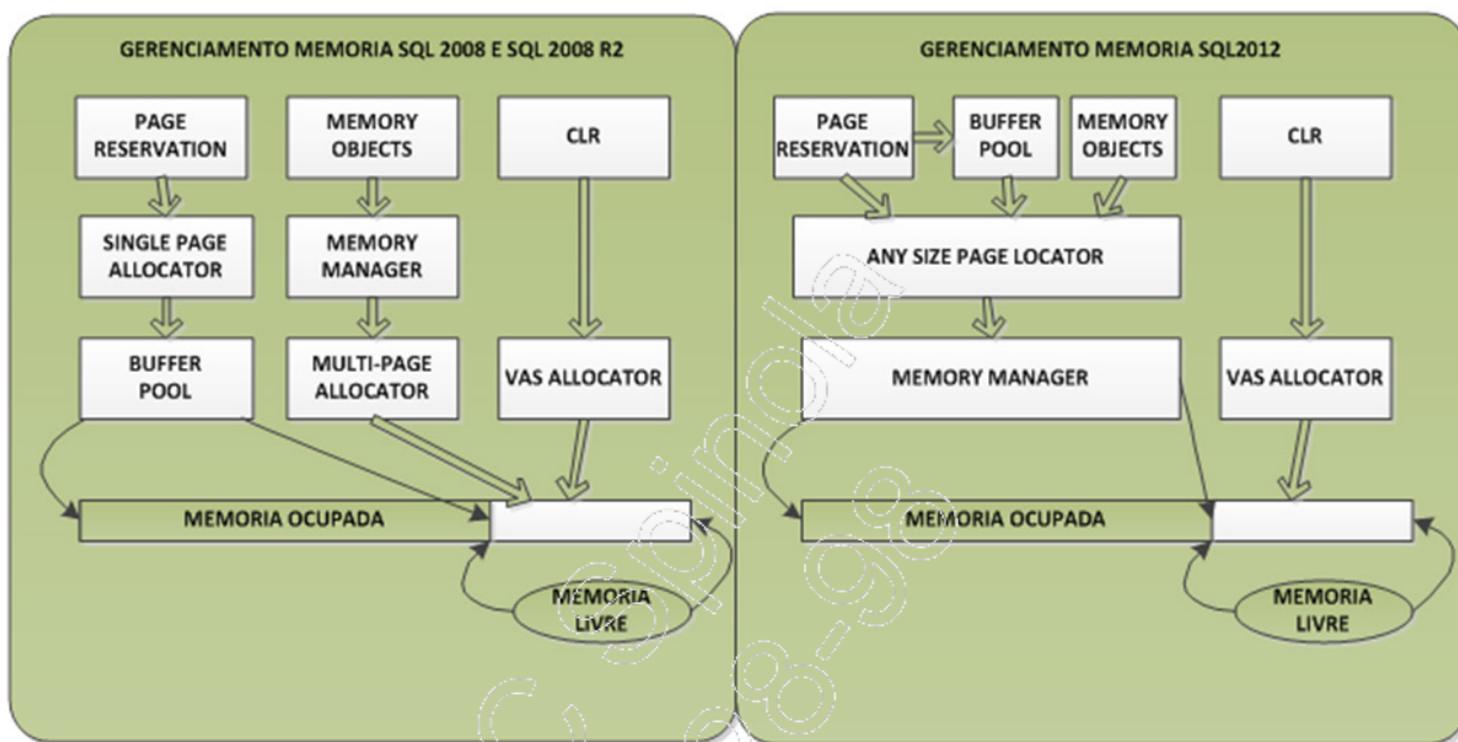
- **AUTO_UPDATE_STATISTICS** (Default **ON**): Caso a opção seja **ON**, haverá a criação automática de estatísticas para complementar a otimização de uma consulta. Se a opção for **OFF**, as estatísticas complementares deverão ser feitas manualmente;
- **AUTO_SHRINK** (Default **OFF**): Caso a opção seja **ON**, haverá a análise para redução periódica dos arquivos de dados e log. O arquivo de log só será reduzido em caso de backup de log (Recuperação **FULL**) ou caso utilize o modo de recuperação **SIMPLE**. Em caso de opção **OFF**, não haverá nenhum tipo de redução;
- **OFFLINE** (Default **false = ONLINE**) | **ONLINE** | **EMERGENCY**: A opção **ONLINE** indica que o banco de dados está aberto e operacional. Já a opção **EMERGENCY** informa que o banco de dados fica no estado de leitura (**READ ONLY**), o processo de geração de log fica desabilitado e apenas usuários membros de **sysadmin** poderão realizar acesso aos bancos de dados. A opção **OFFLINE** indica que o banco de dados está fechado para conexões;
- **READ_ONLY** (Default **false = READ_WRITE**) | **READ_WRITE**: A opção **READ_ONLY** indica que o banco de dados poderá ser apenas consultado e não poderá sofrer nenhuma modificação. A opção **READ_WRITE**, por sua vez, permite a manipulação completa do banco de dados;
- **SINGLE_USER** (Default **false**) | **RESTRICTED_USER** | **MULTI_USER**: A opção **SINGLE_USER** permite que apenas um usuário por vez acesse o banco de dados. Já a opção **RESTRICTED_USER** permite que usuários membros de **sysadmin** e **dbcreator** ou **db_owner** do database possam se conectar. A opção **MULTI_USER**, que é a default, permite que usuários que tenham as devidas permissões possam se conectar à base de dados;

- **RECOVERY:** Para que o SQL Server possa recuperar um banco de dados, existem três opções, as quais influenciam o processo de backup e restore do banco de dados:
 - **FULL:** Esta opção permite a recuperação completa de um banco de dados baseado no backup de dados e no backup dos arquivos de dados. É recomendado em caso de falhas de mídia. Caso os backups sejam realizados, o risco de perda de dados tende a zero;
 - **BULKED_LOGGED:** Neste modelo de recuperação, algumas operações não são gravadas em **LOG**, diminuindo assim a utilização do arquivo de log, recomendado para bancos de dados que sofram carga via **BULKED LOGGED**;
 - **SIMPLE:** Esta opção minimiza a gravação de log, fazendo com que as operações sejam mais rápidas, porém, com um risco de perda de dados muito grande. Executar backups mais regularmente torna-se necessário quando usamos esta opção.
- **PAGE_VERIFY** (Default **checksum**): Esta opção realiza uma operação de verificação da página de dados no momento da criação dessa página. Ela será recomputada e checada contra a informação do checksum armazenado a cada leitura e atualização dela;
- **TORN_PAGE_DETECTION:** Quando uma página for gravada em disco, um bit na parte destinada aos dados será reservado em cada setor de 512 bytes e será armazenado no cabeçalho da página. Quando ela for lida do disco, os bits no cabeçalho serão comparados às informações do setor onde foi copiada a informação do bit;
- **CURSOR_CLOSE_ON_COMMIT** (Default **OFF**): Fecha os cursores que forem abertos quando uma transação for confirmada ou cancelada. Caso a opção seja **OFF**, os cursores permanecerão abertos mesmo quando a transação for confirmada. Em caso de cancelamento da transação, todos os cursores, exceto os definidos como **STATIC** ou **INSENSITIVE** serão fechados;

- **CURSOR_DEFAULT** (Default **GLOBAL**): O escopo de um cursor pode ser **GLOBAL**. Neste caso, podemos usar o cursor em qualquer procedimento ou script TSQL. Já se o valor for **LOCAL**, será usado exclusivamente por um procedimento, gatilho ou TSQL;
- **ALLOW_SNAPSHOT_ISOLATION**: Quando habilitado (**ON**) o nível de isolamento de transação SNAPSHOT poderá ser indicado pelas transações. Desta maneira, qualquer comando verá um SNAPSHOT de dados como existente no início da transação. O valor padrão é **OFF**, o que não permite isolamento de transação SNAPSHOT;
- **READ_COMMITTED_SNAPSHOT**: Esta opção permite o versionamento de linha em vez do bloqueio no nível de isolamento **READ_COMMITTED**. O valor padrão é **OFF**, que utiliza o bloqueio das transações especificadas no nível **READ_COMMITTED**;
- **ENABLE_BROKER**: Habilita o acesso ao Broker do SQL Server;
- **DISABLE_BROKER**: Desabilita o acesso ao Broker do SQL Server;
- **NEW_BROKER**: Cria um novo Broker do SQL Server;
- **ERROR_BROKER_CONVERSATIONS**: Uma mensagem de erro será recebida pelas conversações no banco de dados assim que este for anexado;
- **DB_CHAINING**: Permite (**ON**) que o banco de dados seja fonte ou destino de uma corrente de posse de banco de dados cruzados. **OFF** (Valor padrão) faz com que o banco não faça parte de uma corrente de posse;
- **TRUSTWORTHY**: Caso **ON** seja utilizado, o acesso a recursos localizados externamente ao banco de dados é permitido para bancos que adotem um contexto impersonation. O valor padrão é **OFF**.

2.10. Gerenciamento de memória

O gerenciamento de memória foi alvo de mudanças a partir da versão 2012, com a junção das estruturas chamadas Alocador de Páginas Simples (Single Page Allocator ou SPA) e Alocador de Múltiplas Páginas (Multi-Page Allocator ou MPA).



No diagrama, temos o gerenciamento de memória na versão SQL 2008. Nele constam a estrutura denominada Buffer Pool, responsável pela alocação de dados em memória, e o Multi-page Allocator, para alocação de dados de múltiplas páginas (mais do que uma página). Nesse cenário, tínhamos dois componentes gerenciando a memória, o que, em alguns casos, poderia provocar algum tipo de conflito na alocação da memória. Na versão 2014, o gerenciamento de memória passou a ser centralizado pelo recurso chamado gerenciador de memória (Memory Manager), que foi redesenhado para atender a todas as necessidades de alocação de memória. Isso melhorou o gerenciamento de memória de forma global.

Como exceção ao gerenciamento centralizado de memória, temos a estrutura denominada Espaço de alocação Virtual (Virtual Allocation Space ou VAS). Esta é uma estrutura que gerencia a memória de cada processo (que tem caráter privado). O endereço virtual não é o endereço utilizado em memória por uma página. Em vez disso, são mantidas páginas para cada processo de forma privada. Cada vez que é necessário referenciar um endereço de memória, o sistema realiza uma tradução do endereço virtual em um endereço físico de memória.

Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo.

- Instância é a porção lógica de um banco de dados alocada em memória que tem a finalidade de propiciar o acesso aos bancos de dados e compartilhar recursos de memória, processamento e disco entre eles;
- Banco de dados (Database) é a porção física no gerenciamento de dados, pois tratam dos grupos de arquivos e seus respectivos arquivos de dados e de log;
- A página é a menor unidade de informação do SQL Server. Ela possui o tamanho de 8 KB, sendo que 8060 bytes estão livres para uso;
- Com as extensões, o SQL Server aloca novos espaços para tabelas e índices em um banco de dados. As extensões podem ser uniformes, compostas de 8 páginas (64 KB), ou mistas com até 8 objetos (tabelas e índices podem compartilhar a mesma extensão);
- Uma página não pode ser dividida entre objetos de tipos diferentes. Uma vez alocada para um objeto, somente ele poderá ocupar a respectiva página;
- É importante que o administrador de banco de dados tenha conhecimento e domínio sobre instância, database, FileGroup, extensão, página, pois esses conceitos implicam um armazenamento mais eficiente.

Instância e banco de dados

Teste seus conhecimentos

2

Fernando Cipriola
357.907.3000-0008



IMPACTA
EDITORA

1. Qual banco não é de sistema?

- a) Master
- b) Model
- c) Msdb
- d) TempDB
- e) Recurso

2. Qual a finalidade de um filegroup?

- a) Organizar os arquivos.
- b) Organizar os arquivos de dados.
- c) Organizar os arquivos de dados e log.
- d) Gerenciar dados.
- e) Gerenciar log.

3. O que não é uma característica de um filegroup?

- a) Permite alojar objetos específicos.
- b) Não é possível que um arquivo esteja em mais de um filegroup.
- c) É possível ter mais de um primary.
- d) Pode ser somente leitura.
- e) Não inclui arquivos de log.

4. Qual afirmação está errada sobre página de dados e extensão?

- a) Página de dados corresponde a 8 KB.
- b) Uma extensão corresponde a 18 páginas de dados.
- c) Uma extensão pode ser uniforme ou mista.
- d) Uma extensão uniforme aloca páginas de um único objeto.
- e) Uma extensão corresponde a 8 páginas de dados.

5. Qual(is) o(s) tipo(s) de arquivos que o SQL utiliza?

- a) Dados.
- b) Logs.
- c) Dados e logs.
- d) Dados, logs e índices.
- e) Dados, logs e extensões.

2

Instância e banco de dados

Mãos à obra!

Fernando Spinoza
357.907.



IMPACTA
EDITORA

Observações importantes para os laboratórios

1 - O que será feito nos laboratórios deste capítulo

- Utilização do comando CREATE DATABASE;
- Utilização de objetos do sistema que apresentam informações sobre os databases;
- Utilização das ferramentas gráficas para a criação de databases;
- Utilização do comando ALTER DATABASE;
- Configurações dos databases e dos filegroups;
- Alterações físicas dos files, filegroups e objetos de um database.

2 - Scripts utilizados nestes laboratórios

A pasta **Capítulo_02** contém todos os scripts com os comandos necessários para a realização deste laboratório. Utilize estes scripts para não precisar digitar os respectivos comandos ou apenas para corrigir a execução deles.

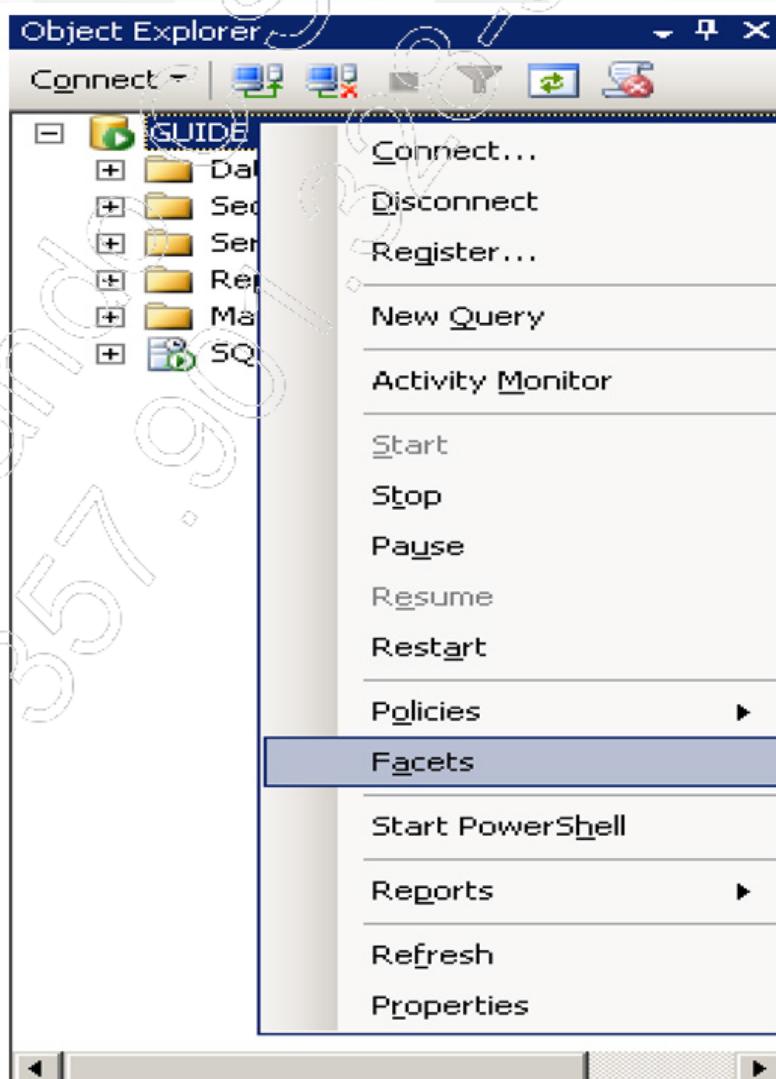
3 - Forma de conexão no sistema operacional

Para realizar os laboratórios deste capítulo, conecte-se ao Windows com a conta de aluno cuja senha é **password**. Para tanto, utilize CTRL + ALT + DEL e escolha a opção **Logoff**. Em seguida, utilize CTRL + ALT + DEL novamente e conecte-se ao Windows com o login **Alunox** (em que x representa o seu número de aluno em sala de aula).

Laboratório 1

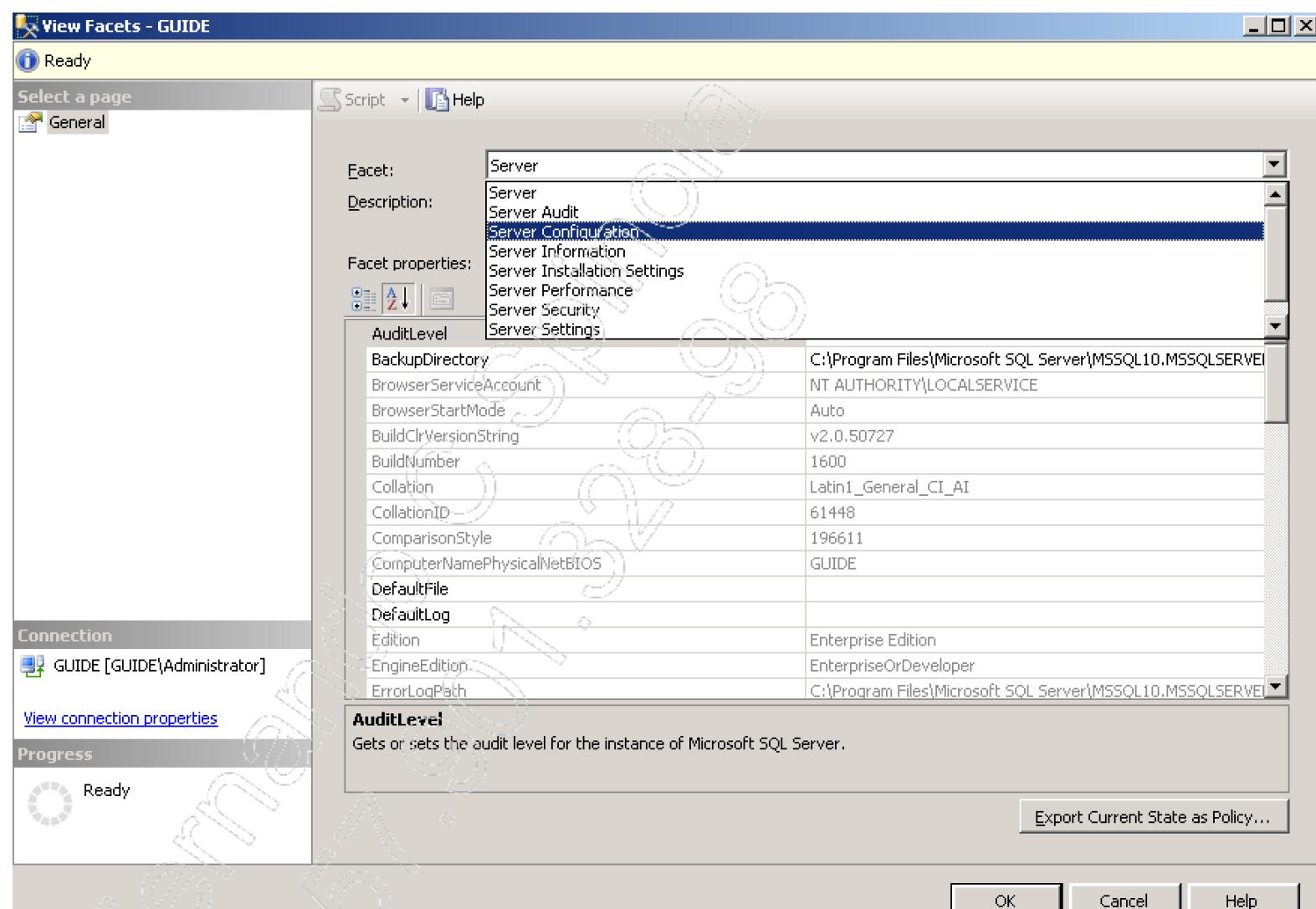
A – Ativando o uso da procedure xp_cmdshell

1. Para ativar o uso da procedure **xp_cmdshell**, clique no botão **Start**;
2. No menu que será exibido, clique em **All Programs**;
3. Selecione **Microsoft SQL Server 2014** e, em seguida, **Microsoft SQL Server 2014 Management Studio**;
4. Na tela **Connect to Server**, escolha a opção **Windows Authentication** para o campo **Authentication** e, no campo **Server Name**, especifique o nome do servidor com o qual será feita a conexão. Clique no botão **Connect**;
5. Dentro do **Object Explorer**, clique com o botão direito do mouse sobre o servidor conectado e, em seguida, selecione **Facets** no menu de contexto:

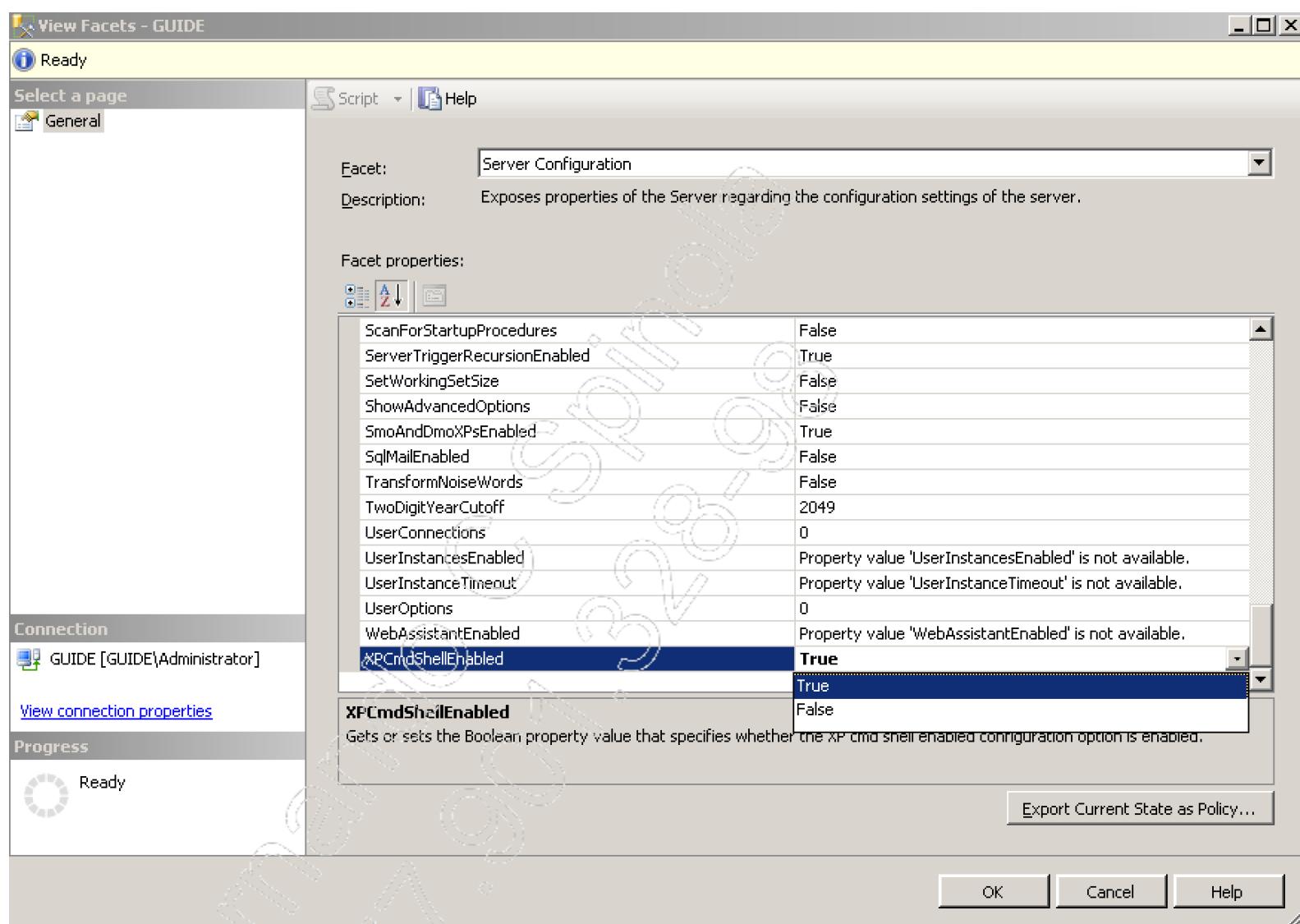


SQL 2014 - Módulo III

6. Será aberta a janela **View Facet**. Na opção **Facet**, selecione o item **Server Configuration**, como mostra a próxima imagem:



7. Na opção **Facet properties**, arraste a barra de rolagem até encontrar o item **XPCmdShellEnabled** e altere o seu valor de **False** para **True**, como mostra a seguinte imagem:



8. Clique em **OK**;

9. Feche a janela **View Facets**.

Laboratório 2

A - Utilizando o comando CREATE DATABASE

- Criando um database com um arquivo de dados e um de log

1. Para a realização deste exercício, analise o quadro a seguir:

Nome do Database: Escola				
Collation: Default				
Script: C:\SQLServer2014 - Módulo III\Capítulo_02\Script_01.sql				
Arquivo(s) de Dado(s)				
Filegroup: Primary				
Nome Lógico	Nome Físico	Tam. Inicial	Tam. Máximo	Crescimento
Escola_Dados	C:\Bancos\Escola\Escola_Dados.mdf	10MB	100MB	10MB
Arquivo(s) de Log(s)				
Nome Lógico	Nome Físico	Tam. Inicial	Tam. Máximo	Crescimento
Escola_Log	C:\Bancos\Escola\Escola_Log.ldf	10MB	10MB	10MB

2. Crie uma pasta **Bancos** com uma subpasta chamada **Escola** na raiz do disco **C:**;

3. Conecte-se ao SQL Server com autenticação do Windows, utilizando a ferramenta **Microsoft SQL Server Management Studio**;

4. Crie o database especificado anteriormente utilizando a linha de comando;

5. Exiba a estrutura física desse database;

6. Exiba a quantidade de **filegroups** desse database;

7. Exiba o **collation** desse database;

8. Exiba informações sobre o **collation** desse database.

Laboratório 3

A – Criando um database com dois arquivos de dados e um de log

- Para realizar este exercício, analise o quadro a seguir:

Nome do Database: Empresa				
Collation: Default				
Script: C:\SQLServer2014 - Módulo III\Capítulo_02\Script_02.sql				
Arquivo(s) de Dado(s)				
Filegroup: Primary				
Nome Lógico	Nome Físico	Tam. Inicial	Tam. Máximo	Crescimento
Empresa_Dados1	C:\Bancos\Empresa\Empresa_Dados1.mdf	10MB	100MB	10MB
Empresa_Dados2	C:\Bancos\Empresa\Empresa_Dados2.ndf	10MB	100MB	10MB
Arquivo(s) de Log(s)				
Empresa_Log	C:\Bancos\Empresa\Empresa_Log.ldf	10MB	10MB	10MB

- Crie uma subpasta chamada **Empresa** na pasta **Bancos**, criada anteriormente na raiz do disco **C:**;
- Conecte-se ao **SQL Server** com autenticação do Windows, utilizando a ferramenta **Microsoft SQL Server Management Studio**;
- Crie o database especificado anteriormente utilizando a linha de comando;

SQL 2014 - Módulo III

5. Crie no database a tabela a seguir:

Nome da Tabela: Funcionario		
Filegroup: Primary		
Nome da Coluna	Datatype	Collation
Cod_Func	Int	
Nome_Func	Varchar(100)	Default

6. Insira os dados a seguir na tabela **Funcionario**:

Cod_Func	Nome_Func
1	Jorge
2	Ronaldo
3	Cláudia
4	Francisco
5	Agnaldo

7. Exiba a estrutura física desse database;
8. Exiba a quantidade de **filegroups** desse database;
9. Exiba o **collation** desse database;
10. Exiba informações sobre o **collation** desse database;
11. Exiba a estrutura na qual se encontra a tabela **Funcionario**.

Laboratório 4

A – Criando um database com três grupos de arquivos (filegroups), dois arquivos dentro do grupo de arquivo dados, um no grupo de arquivo índices e dois arquivos de log

- Para realizar este exercício, analise o quadro a seguir:

Nome do Database: Impacta				
Collation: Default				
Script: C:\SQLServer2014 - Módulo III\Capítulo_02\Script_03.sql				
Arquivo(s) de Dado(s)				
Filegroup: Primary				
Nome Lógico	Nome Físico	Tam. Inicial	Tam. Máximo	Crescimento
Impacta_Dados1	C:\Bancos\Impacta\Impacta_primary1.mdf	10MB	Unlimited	10MB
Filegroup: Tabelas				
Impacta_Dados2	C:\Bancos\Impacta\Impacta_Dados1.ndf	10MB	Unlimited	10MB
Filegroup: Indices				
Impacta_Indices1	C:\Bancos\Impacta\Impacta_indices1.ndf	10MB	Unlimited	10MB
Arquivo(s) de Log(s)				
Nome Lógico	Nome Físico	Tam. Inicial	Tam. Máximo	Crescimento
Impacta_Log1	C:\Bancos\Impacta\Impacta_Log1.ldf	10MB	10MB	10MB
Impacta_Log2	C:\Bancos\Impacta\Impacta_Log2.ldf	10MB	10MB	10MB

SQL 2014 - Módulo III

2. Crie uma subpasta chamada **Impacta** na pasta **Bancos** utilizada anteriormente;
3. Conecte-se ao **SQL Server** com autenticação do Windows, utilizando a ferramenta **Microsoft SQL Server Management Studio**;
4. Crie o database especificado anteriormente utilizando a linha de comando;
5. Crie nesse database as tabelas a seguir:

Nome da Tabela: Aluno		
Filegroup: Tabelas		
Nome da Coluna	Datatype	Collation
Cod_Aluno	Int	
Nome_Aluno	Varchar(100)	Default

Nome da Tabela: Treinamento		
Filegroup: Tabelas		
Nome da Coluna	Datatype	Collation
Cod_Trein	Int	
Nome_Trein	Varchar(100)	Default

6. Insira os dados a seguir na tabela **Aluno**:

Cod_Aluno	Nome_Aluno
1	Cristina
2	Rosana
3	Roberto
4	Marcos
5	Ana

7. Insira os dados a seguir na tabela **Treinamento**:

Cod_Trein	Nome_Trein
1	SQL Server
2	Oracle
3	DB2
4	Sybase
5	Postgresql

8. Exiba a estrutura física desse database;

9. Exiba a quantidade de **filegroups** desse database;

10. Exiba o **collation** deste database;

11. Exiba informações sobre o **collation** desse database;

12. Exiba a estrutura na qual se encontra a tabela **Aluno**;

13. Exiba a estrutura na qual se encontra a tabela **Treinamento**;

14. Exiba o **collation** da coluna **Nome_Aluno** da tabela **Aluno**;

15. Exiba o **collation** da coluna **Nome_Trein** da tabela **Treinamento**;

16. Crie os indices no filegroup **indices**, nas colunas indicadas a seguir:

Nome do Índice	Nome da Tabela	Nome da Coluna Indexada
I_Aluno_1	Aluno	Nome_Aluno
I_Trein_1	Treinamento	Nome_Trein

Laboratório 5

A - Utilizando o comando Microsoft SQL Server Management Studio

- Criando graficamente um database com dois filegroups, um arquivo em cada filegroup e um arquivo de log

1. Para realizar este exercício, analise o quadro a seguir:

Nome do Database: Faculdade				
Collation: Default				
Script: C:\SQLServer2014 - Módulo III\Capítulo_02\Script_04.sql				
Arquivo(s) de Dado(s)				
Filegroup: Primary				
Nome Lógico	Nome Físico	Tam. Inicial	Tam. Máximo	Crescimento
Fac_Dados1	C:\Bancos\faculdade\Fac_Dados1.mdf	10MB	Unlimited	10MB
Filegroup: Indices				
Fac_indices1	C:\Bancos\faculdade\Fac_indices1.ndf	10MB	Unlimited	10MB
Arquivo(s) de Log(s)				
Nome Lógico	Nome Físico	Tam. Inicial	Tam. Máximo	Crescimento
Fac_Log	C:\Bancos\faculdade\Fac_Log.ldf	10MB	10MB	10MB

- Criando os diretórios nos quais os arquivos ficarão armazenados
2. Crie uma subpasta chamada **Faculdade** na pasta **Bancos**, criada anteriormente na raiz do disco C:\;
- Acessando a ferramenta para criar o database graficamente
3. Abra o **Microsoft SQL Server Management Studio**;
4. Conecte-se com a autenticação do Windows;
5. Clique com o botão direito do mouse sobre a pasta **Databases**;
6. Escolha a opção **New Database**;
- Criando um novo Filegroup no Database
7. Na tela que será exibida, clique na opção **Filegroup**;
8. Clique no botão **Add** para acrescentar um outro **Filegroup** nesse database;
9. No campo **Name**, escreva a palavra **Índice**;
- Descrevendo o Database
10. Clique na opção **General**;
11. No campo **Database Name**, escreva a palavra **Faculdade**;
- Descrevendo o primeiro arquivo de dados
12. No campo **Logical Name**, escreva **Fac_Dados1**;
13. No campo **Initial Size**, escreva 10;
14. No campo **Autogrowth**, clique nas reticências (...). Podemos observar que é default a opção de autocrescimento no valor de 10MB, com tamanho máximo **Unrestricted File Growth**;
15. Clique no botão **OK**;

SQL 2014 - Módulo III

16. Arraste a barra de rolagem horizontal para a direita e, no campo **Path**, escreva **C:\Bancos\Faculdade**;

- **Descrevendo o arquivo de log**

17. No campo **Logical Name** da linha que se refere ao arquivo de Log, no lugar da palavra **Faculdade_Log**, escreva **Fac_Log**;

18. No campo **Autogrowth**, clique nas reticências (...);

19. Na tela que será exibida, escolha a opção **In Megabyte** e, no campo ao lado desta opção, escreva **10**;

20. Clique no botão **OK**;

21. Mova a barra de rolagem horizontal para a direita e, no campo **Path**, escreva **C:\Bancos\faculdade**;

- **Descrevendo o segundo arquivo de dados**

22. Clique no botão **Add**;

23. No campo **Logical Name**, escreva **Fac_Dados2**;

24. No campo **Filegroup**, escolha a opção **Indice**;

25. No campo **Initial Size**, escreva **10**;

26. No campo **Autogrowth**, clique nas reticências;

27. Na tela que será exibida, no campo ao lado da opção **In Megabyte**, escreva **10**;

28. Clique no botão **OK**;

29. No campo **Path**, escreva **C:\Bancos\Faculdade**;

30. Clique no botão **OK**.

Laboratório 6

A – Utilizando o comando ALTER DATABASE

1. O esquema a seguir exibe a estrutura do database **Universidade** com suas respectivas tabelas. Para a realização deste exercício, acesse a ferramenta **Microsoft SQL Server Management Studio**, abra uma nova **Query** e execute o código para a criação do banco e solução dos exercícios. Para tanto, utilize o script **C:\SQLServer2014 - Módulo III\Capítulo_02\Script_05.sql**;

Database	Universidade	
Filegroup	Primary	Dados
Arquivos	Univers_primary.mdf	Univers_Dados1.ndf
Tabelas	Aluno	Funcionario

2. Coloque o database **Universidade** em uso;
3. Crie as tabelas: **Aluno** e **Funcionario**;
4. Insira os dados nas tabelas conforme o script;
5. Marque o database **Universidade** como **Read-Only**;
6. Acesse o database **Universidade** e tente inserir um registro na tabela **Aluno**;
7. Acesse o database **Master** e marque o database **Universidade** como **Offline**;
8. Tente acessar o database **Universidade**;
9. No database **Master**, marque o database **Universidade** como **On-line**;
10. Marque o database **Universidade** como **Read-Write**;
11. Acesse o database **Universidade**;
12. Marque o filegroup **DADOS** como **Read-Only**;
13. Tente inserir um registro de dados na tabela **Funcionario**;

SQL 2014 - Módulo III

14. Marque o filegroup **DADOS** como **Read-Write**;
15. Tente inserir um registro na tabela **Funcionario**;
16. Configure o filegroup **DADOS** como **Default**;
17. No database **Universidade**, crie a tabela **Materia** especificada a seguir sem apontar para ela nenhum filegroup;

Nome da Tabela: Materia	
Nome da Coluna	Datatype
Cod_Mat	Int
Nome_Mat	Varchar(50)

18. Verifique em qual filegroup essa tabela foi criada;
19. Configure o filegroup **Primary** para ser novamente o default;
20. Acrescente no database **Universidade** mais um arquivo no filegroup **DADOS_BASICO**;
21. Acrescente ao filegroup **DADOS_BASICO** um arquivo que deverá chamar-se **Univers_Dados3.ndf**, com os atributos especificados a seguir:

Nome Lógico	Nome Físico	Tam. Inicial	Tam. Máximo	Crescimento
Univers_Dados3	C:\Bancos\Universidade\Univers_Dados3.NDF	10MB	Unlimited	10MB

22. Acrescente no database **Universidade** o filegroup chamado **DADOS_COMPLEMENTO**;

23. Acrescente no filegroup **DADOS_COMPLEMENTO** dois arquivos de dados chamados **Univers_Dados_complemento1.ndf** e **Univers_Dados_complemento2.ndf**, cujos atributos estão especificados a seguir:

Nome Lógico	Nome Físico	Tam. Inicial	Tam. Máximo	Crescimento
Univers_Dados4	C:\Bancos\Universidade\Univers_Dados_complemento1.ndf	10MB	Unlimited	10MB
Univers_Dados5	C:\Bancos\Universidade\Univers_Dados_complemento2.ndf	10MB	Unlimited	10MB

24. No filegroup **DADOS_BASICO**, crie a tabela **Professor** conforme estrutura exibida a seguir:

Nome da Tabela: Professor	
Nome da Coluna	Datatype
Cod_Prof	Int
Nome_Prof	Varchar(50)

25. Exiba a estrutura dos Filegroups e arquivos do database **Universidade**:

SQL 2014 - Módulo III

26. Altere os seguintes parâmetros do arquivo **Univers_Dados5**:

New Name	Univers_Dados6
Size	Acrescentar 10MB
MaxSize	Mudar para Unlimited
Filegrowth	Passar para 20MB

27. Diminua 5MB do tamanho do arquivo **Univers_Dados6**;

28. Elimine o arquivo **Univers_Dados6** do filegroup **dados_complemento**;

29. Elimine o arquivo **Univers_Dados4** do filegroup **dados_complemento**;

30. Elimine o filegroup **dados_complemento** desse database;

31. Exiba a estrutura do database **Universidade**.

Gerenciando tabelas e outros objetos

3

- ✓ Criando tabelas;
- ✓ Visões;
- ✓ Criando sequências;
- ✓ Criando sinônimos;
- ✓ Criando tipos.



IMPACTA
EDITORA

3.1. Introdução

O banco de dados SQL Server permite a criação de diversos tipos de tabelas. A gestão dessas tabelas cabe ao administrador de banco de dados, por isso, é fundamental compreender cada tipo de tabela, seus usos, benefícios e consequências da sua utilização. Neste capítulo, abordaremos amplamente esse assunto. Também vamos tratar de algumas práticas que são importantes para otimizar a criação desse tipo de objeto.

Devemos entender que as tabelas podem ter finalidades diferentes. Em relação ao tempo, as tabelas podem ser permanentes ou temporárias. Elas ainda podem ser inteiras ou particionadas. Essa prática de particionamento pode ser útil no caso de tabelas com grandes quantidades de dados (milhões de registros).

3.2. Criando tabelas

Existem vários tipos de tabelas em um banco de dados SQL Server:

- Tabelas regulares (REGULAR TABLE);
- Tabelas temporárias locais;
- Tabelas temporárias globais;
- Tabelas baseadas em consulta;
- Tabelas particionadas;
- Tabelas baseadas em arquivos (FILETABLE).

3.2.1. Constraints

Para realizarmos operações de manutenção, recriação e cópia de bancos de dados, é importante entendermos os objetivos das constraints na criação de tabelas do banco de dados.

Constraints são regras de integridade referencial impostas a um banco de dados na sua criação e manutenção. Essas regras valem contra toda e qualquer aplicação que manipule o banco em operações de comandos de manipulação de dados (DML).

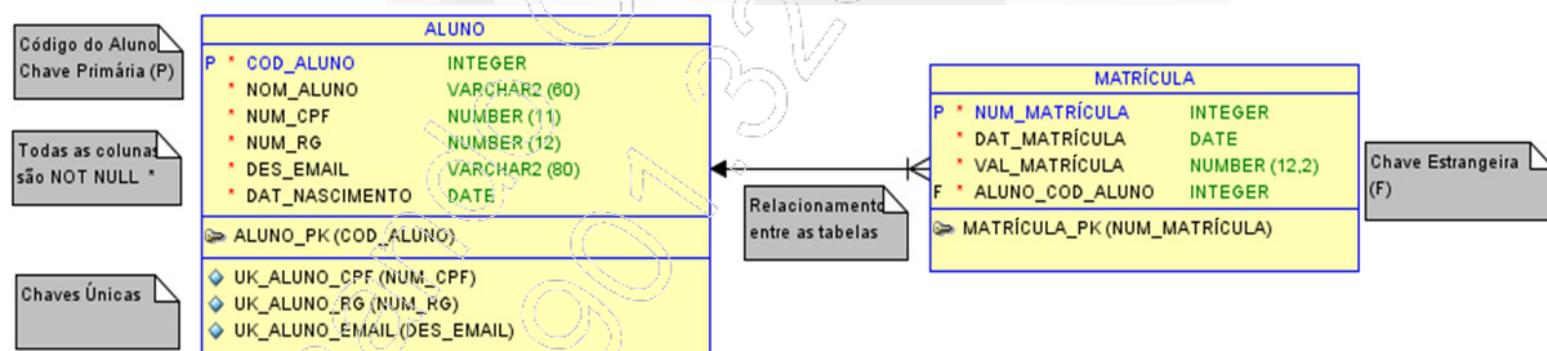
Existem cinco constraints de banco de dados, cada uma com finalidades específicas:

- **Chave primária:** Garante que essa chave não tenha valores duplicados. Em função da modelagem de dados e do relacionamento existente entre as tabelas, é necessário que a tabela possua chave primária. Uma (chave primária simples) ou mais colunas (chave primária composta) podem fazer parte da chave primária. Neste caso, a soma dos valores de todas as colunas representa a chave primária por si mesma. Também é conhecida como Primary Key ou ainda PK. Geralmente as tabelas possuem uma coluna que as representa, mas, caso essa coluna não exista, é possível criá-la. As colunas devem ser sempre listadas como as primeiras de uma tabela, por questões de organização física nas páginas de dados. Toda chave primária é representada por um índice único, que faz o suporte que garante que os dados não serão duplicados. Ela deve ser criada ao mesmo tempo ou depois que a tabela, porém, antes das chaves estrangeiras das tabelas;

- **Chave única:** A chave única é a segunda regra de integridade de dados (Constraint). Ela não é obrigatória e serve como uma garantia de não duplicidade de uma informação, além disso, a chave única é uma forma de permitir pesquisas mais precisas, como no caso das chaves primárias. Há muita confusão entre essas duas chaves, porém, para simplificar, devemos entender que a chave única é um mecanismo eficiente e rápido de localização de dados apenas. As chaves únicas também são conhecidas como chaves alternadas ou ainda Unique Key ou UK. Para exemplificar, imaginemos uma tabela que contenha dados de alunos. É natural relacionar essa informação com suas matrículas em determinados cursos, logo há uma relação entre matrículas e alunos. Dessa forma, o código do aluno pode ser considerado como chave primária da tabela de alunos, porém, os alunos têm registro geral (RG), que é uma informação única. Se a pessoa não souber seu código de aluno, com certeza saberá seu RG. Então, podemos usar essa informação como chave única. Em uma tabela, é possível ter uma ou mais colunas e várias chaves únicas. Por exemplo, se o aluno tiver como informação o CPF, passamos a ter duas chaves únicas individuais, RG e/ou CPF, além da chave primária que é o código do aluno;
- **Check:** Constraint de check é a regra que garante que uma determinada coluna terá apenas valores específicos, impedindo que valores indesejados sejam atribuídos a ela. Por exemplo, temos a coluna sexo, na qual são esperados dois valores: M para representar o sexo masculino, F para o feminino e mais nenhum outro valor além desses dois. Cada tabela pode conter várias colunas com constraints de check;
- **Not Null:** Constraint de obrigatoriedade de valores. A coluna submetida a esta constraint torna obrigatório que sejam informados valores para as colunas ligadas a esta constraint. Toda tabela pode ter desde nenhuma até todas as colunas obrigatórias. Todas as colunas de chave primária e ou única precisam ter adicionalmente esta constraint definida, logo essas colunas terão duas constraints ao mesmo tempo;

- Chave estrangeira:** Esta constraint indica que uma tabela está ligada a outra. Para isso, é necessário entender que, antes, a constraint de chave primária precisa estar ativa na tabela de origem da informação. No exemplo citado anteriormente, uma matrícula está ligada a um aluno, logo, sem aluno, não há matrícula. Para que esta relação ocorra, uma coluna com o mesmo tipo de dado (da coluna do código do aluno da tabela de aluno) e tamanho deve ser colocada na tabela de matrícula. Já o nome desta coluna pode ser até o mesmo da coluna do código da tabela de aluno. Com esse requisito cumprido, podemos criar a ligação entre as tabela e essa constraint tem essa finalidade. Isso irá garantir que, se cadastrarmos uma matrícula com um código de aluno, este deverá estar cadastrado na tabela de alunos. Além disso, com a matrícula localizada, conseguiremos obter as informações de um aluno.

Vejamos este exemplo esboçado no esquema a seguir:



3.2.2. Tipos de dados SQL Server (DataTypes)

Os tipos de dados são muito importantes para a criação de tabelas no SQL Server. Conheceremos cada tipo de dado a seguir:

Tipo Dado	Menor Valor	Maior Valor	Armazenamento	Tipo	Notas
Bigint	-9,22337E+18	2^63-1	8 bytes	Exato	
Int	-2,147,483,648	2,147,483,647	4 bytes	Exato	
Smallint	-32,768	32,767	2 bytes	Exato	
Tinyint	0	255	1 bytes	Exato	
Bit	0	1	1 a 8 bits	Exato	
Decimal	1E+38	10^38-1	1-9 ≈ 5 bytes 10 a 19 = 9 bytes 20-28 = 13 bytes 29-38 = 17 bytes	Exato	
Numeric	no				
Money	-9,22337E+14	2^63-1 / 10000	8 bytes	Exato	
Smallmoney	-214,748.3648	214,748.3647	4 bytes	Exato	
Float	-1.79E + 308	1.79E + 308	4 bytes até 25 posições e 8 bytes entre 26 e 53 posições	Aproximado	De 1 a 53 casas decimais
Real	-3.40E + 38	3.40E + 38	4 bytes	Aproximado	7 casas decimais

Gerenciando tabelas e outros objetos



Tipo Dado	Menor Valor	Maior Valor	Armazenamento	Tipo	Notas
Datetime	1753-01-01 00:00:00.000	9999-12-31 23:59:59.997	8 bytes	Datetime	Precisão de milissegundos
Smalldatetime	01/01/1900 00:00	06/06/2079 23:59		Datetime	
Date	0001-01-01	31/12/9999		Datetime	
Time	00:00:0000000	23:59:59.9999999		Datetime	Usando TIME(3), precisão de milissegundos é TIME(7), maior nível de precisão possível
Datetime2	0001-01-01 00:00:0000000	9999-12-31 23:59:59.9999999	6 a 8 bytes	Datetime	Combina date e time
Datetimeoffset	0001-01-01 00:00:0000000 -14:00	9999-12-31 23:59:59.9999999 +14:00	Precisão 8 a 10 bytes	Datetime	Combina date + time + offset
Char	0 chars	8000 caracteres	Mesmo tamanho definido para coluna	Caracter	Tam. Fixo

SQL 2014 - Módulo III

Tipo Dado	Menor Valor	Maior Valor	Armazenamento	Tipo	Notas
Varchar	0 caracteres	8000 caracteres	Número de caracteres utilizados + 2 bytes	Caracter	Tam. Variável
Varchar(max)	0 caracteres	2^{31} caracteres	Número de caracteres utilizados + 2 bytes	Caracter	Tam. Variável
Text	0 caracteres	2,147,483,647 caracteres	Número de caracteres utilizados + 4 bytes	Caracter	Tam. Variável
Nchar	0 caracteres	4000 caracteres	Tamanho definido x 2	Unicode	Tam. Fixo
Nvarchar	0 caracteres	4000 caracteres		Unicode	Tam. Variável
Nvarchar(max)	0 caracteres	2^{30} caracteres		Unicode	Tam. Variável
Ntext	0 caracteres	1,073,741,823 caracteres		Unicode	Tam. Variável
Binario	0 bytes	8000 bytes		Binário	Tam. Fixo
VarBinario	0 bytes	8000 bytes		Binário	Tam. Variável
VarBinario(max)	0 bytes	2^{31} bytes		Binário	Tam. Variável

Tipo Dado	Menor Valor	Maior Valor	Armazenamento	Tipo	Notas
Image	0 bytes	2,147,483,647 bytes		Binário	
SqL_variant				Outro	
Timestamp				Outro	
Uniqueidentifier				Outro	Armazena o identificador global (GUID)
Xml				Outro	Armazena dados XML
Cursor				Outro	Referência para Cursor
Table				Outro	Armazena o resultado de uma consulta para acesso posterior

3.2.3. Tabelas regulares

As tabelas regulares são as tabelas criadas normalmente por meio do comando CREATE TABLE.

Tabelas regulares podem ser criadas com ou sem as constraints, como veremos a seguir:

- Exemplo de tabela sem nenhuma constraint criada no grupo de arquivo (FILEGROUP) **Primary**:

```
CREATE TABLE ALUNO
(
    COD_ALUNO INTEGER ,
    NOM_ALUNO VARCHAR (60) ,
    NUM_CPF NUMERIC (11) ,
    NUM_RG NUMERIC (12) ,
    DES_EMAIL VARCHAR (80) ,
    DAT_NASCIMENTO DATETIME
)
```

- Exemplo de tabela sem nenhuma constraint criada no grupo de arquivo (FILEGROUP) **DADOS**:

```
CREATE TABLE ALUNO
(
    COD_ALUNO INTEGER ,
    NOM_ALUNO VARCHAR (60) ,
    NUM_CPF NUMERIC (11) ,
    NUM_RG NUMERIC (12) ,
    DES_EMAIL VARCHAR (80) ,
    DAT_NASCIMENTO DATETIME
) ON DADOS
```

- Exemplo de tabela com constraint de **not null**:

```
CREATE TABLE ALUNO
(
    COD_ALUNO INTEGER NOT NULL ,
    NOM_ALUNO VARCHAR (60) NOT NULL ,
    NUM_CPF NUMERIC (11) NOT NULL ,
    NUM_RG NUMERIC (12) NOT NULL ,
    DES_EMAIL VARCHAR (80) NOT NULL ,
    DAT_NASCIMENTO DATETIME NOT NULL
)
ON DADOS
```

- Exemplo de criação de tabela com constraint de chave primária nomeada e campos **not null**:

```
CREATE TABLE ALUNO
(
    COD_ALUNO INTEGER NOT NULL ,
    NOM_ALUNO VARCHAR (60) NOT NULL ,
    NUM_CPF NUMERIC (11) NOT NULL ,
    NUM_RG NUMERIC (12) NOT NULL ,
    DES_EMAIL VARCHAR (80) NOT NULL ,
    DAT_NASCIMENTO DATETIME NOT NULL ,
    CONSTRAINT ALUNO_PK PRIMARY KEY CLUSTERED (COD_ALUNO)
)
ON DADOS
```

Vejamos o mesmo resultado, porém, agora a constraint foi declarada ao mesmo tempo que a coluna:

```
CREATE TABLE ALUNO
(
    COD_ALUNO INTEGER NOT NULL CONSTRAINT ALUNO_PK PRIMARY KEY
CLUSTERED,
    NOM_ALUNO VARCHAR (60) NOT NULL ,
    NUM_CPF NUMERIC (11) NOT NULL ,
    NUM_RG NUMERIC (12) NOT NULL ,
    DES_EMAIL VARCHAR (80) NOT NULL ,
    DAT_NASCIMENTO DATETIME NOT NULL)
ON DADOS
```

Esta opção só é válida se a coluna de chave primária for simples. Caso tenhamos uma chave primária composta de duas colunas, esta forma não poderá ser usada.

Note que a constraint de **not null** ganhou um nome (**ALUNO_PK**).

- Sintaxe da constraint de chave primária nomeada:

```
CONSTRAINT nome_constraint PRIMARY KEY CLUSTERED (nome  
coluna[, nome coluna2[,...]])
```

- Exemplo de criação de tabela com constraint de chave primária não nomeada e campos **not null**:

```
CREATE TABLE ALUNO  
(  
    COD_ALUNO INTEGER NOT NULL PRIMARY KEY ,  
    NOM_ALUNO VARCHAR (60) NOT NULL ,  
    NUM_CPF NUMERIC (11) NOT NULL ,  
    NUM_RG NUMERIC (12) NOT NULL ,  
    DES_EMAIL VARCHAR (80) NOT NULL ,  
    DAT_NASCIMENTO DATETIME NOT NULL  
)  
ON DADOS
```

Nesse exemplo, não estamos nomeando a chave primária, logo, caso ocorra tentativa de violação da chave primária de aluno, não será mostrado o nome da constraint violada. É recomendável, para uma administração de banco de dados mais eficiente, que as constraints tenham nomes claros.

- Exemplo de criação de tabela com constraint de chave primária nomeada, campos **not null** e índice da chave primária em outro grupo de arquivos (FILEGROUP):

```
CREATE TABLE ALUNO
(
    COD_ALUNO INTEGER NOT NULL ,
    NOM_ALUNO VARCHAR (60) NOT NULL ,
    NUM_CPF NUMERIC (11) NOT NULL ,
    NUM_RG NUMERIC (12) NOT NULL ,
    DES_EMAIL VARCHAR (80) NOT NULL ,
    DAT_NASCIMENTO DATETIME NOT NULL ,
    CONSTRAINT ALUNO_PK PRIMARY KEY CLUSTERED (COD_ALUNO)
    WITH (
        ALLOW_PAGE_LOCKS = ON ,
        ALLOW_ROW_LOCKS = ON )
    ON INDICES
)
ON DADOS
GO

CREATE TABLE MATRICULA
(
    NUM_MATRICULA INTEGER NOT NULL ,
    DAT_MATRICULA DATETIME NOT NULL ,
    VAL_MATRICULA DECIMAL (12,2) NOT NULL ,
    ALUNO_COD_ALUNO INTEGER NOT NULL ,
    CONSTRAINT MATRICULA_PK PRIMARY KEY CLUSTERED (NUM_MATRICULA)
    WITH (
        ALLOW_PAGE_LOCKS = ON ,
        ALLOW_ROW_LOCKS = ON )
    ON INDICES
)
ON DADOS
GO
```

SQL 2014 - Módulo III

- Exemplo de criação de chave única (primeira forma):

```
CREATE UNIQUE NONCLUSTERED INDEX
    UK_ALUNO_CPF ON ALUNO
    (
        NUM_CPF
    )
    ON INDICES
GO

CREATE UNIQUE NONCLUSTERED INDEX
    UK_ALUNO_RG ON ALUNO
    (
        NUM_RG
    )
    ON INDICES
GO

CREATE UNIQUE NONCLUSTERED INDEX
    UK_ALUNO_EMAIL ON ALUNO
    (
        DES_EMAIL
    )
    ON INDICES
GO
```

- Exemplo de criação de chave única (segunda forma):

```
CREATE TABLE ALUNO
(
    COD_ALUNO INTEGER NOT NULL ,
    NOM_ALUNO VARCHAR (60) NOT NULL ,
    NUM_CPF NUMERIC (11) NOT NULL ,
    NUM_RG NUMERIC (12) NOT NULL ,
    DES_EMAIL VARCHAR (80) NOT NULL ,
    DAT_NASCIMENTO DATETIME NOT NULL ,
    CONSTRAINT UK_ALUNO_CPF UNIQUE (NUM_CPF),
    CONSTRAINT UK_ALUNO_RG UNIQUE (NUM_RG),
    CONSTRAINT UK_ALUNO_EMAIL UNIQUE (DES_EMAIL),
    CONSTRAINT ALUNO_PK PRIMARY KEY CLUSTERED (COD_ALUNO)
    WITH (
        ALLOW_PAGE_LOCKS = ON ,
        ALLOW_ROW_LOCKS = ON )
    ON INDICES
)
ON DADOS
```

- Exemplo de criação de chave única (terceira forma):

```
CREATE TABLE ALUNO
(
    COD_ALUNO INTEGER NOT NULL ,
    NOM_ALUNO VARCHAR (60) NOT NULL ,
    NUM_CPF NUMERIC (11) NOT NULL CONSTRAINT UK_ALUNO_CPF UNIQUE ,
    NUM_RG NUMERIC (12) NOT NULL CONSTRAINT UK_ALUNO_RG UNIQUE ,
    DES_EMAIL VARCHAR (80) NOT NULL CONSTRAINT UK_ALUNO_EMAIL UNIQUE,
    DAT_NASCIMENTO DATETIME NOT NULL ,
    CONSTRAINT ALUNO_PK PRIMARY KEY CLUSTERED (COD_ALUNO)
    WITH (
        ALLOW_PAGE_LOCKS = ON ,
        ALLOW_ROW_LOCKS = ON )
    ON INDICES
)
ON DADOS
```

- Exemplo de criação de tabela com constraint de chave primária nomeada, campos **not null** e índice da chave primária em outro grupo de arquivos (FILEGROUP) e chave estrangeira:

```
CREATE TABLE MATRICULA
(
    NUM_MATRICULA INTEGER NOT NULL ,
    DAT_MATRICULA DATETIME NOT NULL ,
    VAL_MATRICULA DECIMAL (12,2) NOT NULL ,
    COD_ALUNO INTEGER NOT NULL ,
    FOREIGN KEY (COD_ALUNO) REFERENCES ALUNO(COD_ALUNO),
    CONSTRAINT MATRICULA_PK PRIMARY KEY CLUSTERED (NUM_MATRICULA)
    WITH (
        ALLOW_PAGE_LOCKS = ON ,
        ALLOW_ROW_LOCKS = ON )
    ON INDICES
)
ON DADOS
GO
```

SQL 2014 - Módulo III

- Exemplo de criação de tabela com constraint de chave primária nomeada, campos **not null** e índice da chave primária em outro grupo de arquivos (FILEGROUP), chave estrangeira e constraint de check:

```
CREATE TABLE MATRICULA
(
    NUM_MATRICULA INTEGER NOT NULL ,
    DAT_MATRICULA DATETIME NOT NULL ,
    VAL_MATRICULA DECIMAL (12,2) NOT NULL ,
    COD_ALUNO INTEGER NOT NULL ,
    FOREIGN KEY (COD_ALUNO) REFERENCES ALUNO(COD_ALUNO) ,
    CHECK(VAL_MATRICULA > 0),
    CONSTRAINT MATRICULA_PK PRIMARY KEY CLUSTERED (NUM_MATRICULA)
    WITH (
        ALLOW_PAGE_LOCKS = ON ,
        ALLOW_ROW_LOCKS = ON )
        ON INDICES
)
ON DADOS
GO
```

- Segunda forma de criar chave estrangeira:

```
CREATE TABLE MATRICULA
(
    NUM_MATRICULA INTEGER NOT NULL ,
    DAT_MATRICULA DATETIME NOT NULL ,
    VAL_MATRICULA DECIMAL (12,2) NOT NULL ,
    COD_ALUNO INTEGER NOT NULL FOREIGN KEY REFERENCES ALUNO(COD_ALUNO) ,
    CHECK(VAL_MATRICULA > 0),
    CONSTRAINT MATRICULA_PK PRIMARY KEY CLUSTERED (NUM_MATRICULA)
    WITH (
        ALLOW_PAGE_LOCKS = ON ,
        ALLOW_ROW_LOCKS = ON )
        ON INDICES
)
ON DADOS
GO
```

Para alterar uma tabela, existe o comando **ALTER TABLE**. Com ele, podemos realizar as seguintes ações:

- Adicionar constraints;
- Adicionar nova coluna;
- Alterar coluna existente;
- Eliminar uma coluna;
- Habilitar constraint;
- Desabilitar constraint;
- Eliminar constraint;
- Renomear uma tabela.

Exemplos:

- Adicionando uma nova coluna em uma tabela:

```
ALTER TABLE ALUNO ADD DAT_CADASTRO DATETIME;
```

- Adicionando mais de uma nova coluna em uma tabela:

```
ALTER TABLE ALUNO ADD DAT_CADASTRO DATETIME, STA_ATIVO CHAR(1) NOT NULL
```

- Adicionando uma constraint (chave primária, única e estrangeira) em uma tabela:

```
ALTER TABLE ALUNO ADD PRIMARY KEY (COD_ALUNO)
ALTER TABLE ALUNO ADD UNIQUE (DES_EMAIL)
ALTER TABLE MATRICULA ADD FOREIGN KEY (COD_ALUNO) REFERENCES ALUNO
(COD_ALUNO)
```

Ou:

```
ALTER TABLE ALUNO ADD CONSTRAINT ALUNO_PK PRIMARY KEY (COD_ALUNO)
ALTER TABLE ALUNO ADD CONSTRAINT UK_ALUNO_EMAIL UNIQUE (DES_EMAIL)
ALTER TABLE MATRICULA ADD CONSTRAINT FK_ALUNO_MATRICULA FOREIGN
KEY (COD_ALUNO) REFERENCES ALUNO (COD_ALUNO)
```

SQL 2014 - Módulo III

- Eliminando uma chave primária, única ou estrangeira:

```
ALTER TABLE ALUNO DROP CONSTRAINT ALUNO_PK  
ALTER TABLE ALUNO DROP CONSTRAINT UK_ALUNO_EMAIL  
ALTER TABLE MATRICULA DROP CONSTRAINT FK_ALUNO_MATRICULA
```

- Para desabilitar qualquer constraint específica:

```
ALTER TABLE MATRICULA NOCHECK CONSTRAINT FK_ALUNO_MATRICULA
```

- Para desabilitar todas as constraints de uma tabela:

```
ALTER TABLE MATRICULA NOCHECK CONSTRAINT ALL
```

- Para habilitar uma constraint específica:

```
ALTER TABLE MATRICULA CHECK CONSTRAINT FK_ALUNO_MATRICULA
```

- Para habilitar todas as constraints em uma tabela:

```
ALTER TABLE MATRICULA CHECK CONSTRAINT ALL
```

- Para eliminar uma coluna da tabela:

```
ALTER TABLE ALUNO DROP COLUMN DAT_CADASTRO
```

- Para renomear uma tabela, o comando não é ALTER e sim SP_RENAME:

```
EXEC SP_RENAME 'MATRICULA', 'MATRICULA_ALUNO'
```

Ou:

```
EXEC SP_RENAME 'MATRICULA_ALUNO.NUM_MATRICULA', 'ID_MATRICULA' ,  
'COLUMN'
```

- Para eliminar uma tabela:

```
DROP TABLE MATRICULA
```

- Para visualizar a estrutura da tabela:

```
EXEC SP_HELP 'MATRICULA'
```

- Listando as tabelas em um banco de dados:

```
SELECT name, [type], type_desc FROM sys.objects
WHERE schema_Name(schema_id) = 'dbo'
AND type IN ('U', 'V', 'FN', 'TF', 'TT', 'P')
--U - User Table
--V - View
--FN - Scalar Valued Function
--TF - Table Valued Function
--TT - Table Type
--P - Stored Procedure
--I - indexes
```

3.2.4. Tabelas temporárias locais

Tabelas temporárias têm como principal característica a visibilidade apenas para o usuário que a criou e durante a conexão vigente. Elas são eliminadas após a desconexão do usuário e utilizam o banco de dados **tempdb**. As tabelas temporárias locais podem utilizar o mecanismo de constraints, assim como as tabelas permanentes. Para criarmos uma tabela temporária local, basta adicionar o caractere # no início do nome da tabela.

Exemplo:

```
CREATE TABLE #MATRICULA
(
    NUM_MATRICULA INTEGER NOT NULL ,
    DAT_MATRICULA DATETIME NOT NULL ,
    VAL_MATRICULA DECIMAL (12,2) NOT NULL ,
    COD_ALUNO INTEGER NOT NULL,
    CONSTRAINT MATRICULA_PK PRIMARY KEY CLUSTERED (NUM_MATRICU-
LA)

)
```

3.2.5. Tabelas temporárias globais

Tabelas temporárias globais têm como principal característica a visibilidade para todos os usuários. Estas são eliminadas após a desconexão do usuário. As tabelas temporárias globais podem utilizar o mecanismo de constraints, assim como as tabelas permanentes. Para criarmos uma tabela temporária global, basta adicionar o caractere ## no início do nome da tabela. Exemplo:

```
CREATE TABLE ##MATRICULA
(
    NUM_MATRICULA INTEGER NOT NULL ,
    DAT_MATRICULA DATETIME NOT NULL ,
    VAL_MATRICULA DECIMAL (12,2) NOT NULL ,
    COD_ALUNO INTEGER NOT NULL,
    CONSTRAINT MATRICULA_PK PRIMARY KEY CLUSTERED (NUM_MATRICU-
LA)

)
```

3.2.6. Tabelas baseadas em consultas

Tabelas baseadas em consultas são essencialmente tabelas geradas e carregadas durante a execução de uma consulta. Para isso, a tabela gerada não pode ter sido previamente criada, logo, não pode existir. Ela é carregada através do comando **SELECT ... INTO**. Sua origem poderá ser uma tabela única, ou a junção de várias tabelas, ou ainda uma ou mais visões. Essas tabelas são criadas sem o mecanismo de constraint, mesmo que nas tabelas originais existam constraints. Esse tipo de tabela pode ser permanente ou temporária.

Exemplo de criação de tabela baseada em consulta:

```
Select * into ALUNO_BACKUP from ALUNO
```

3.2.7. Tabelas particionadas

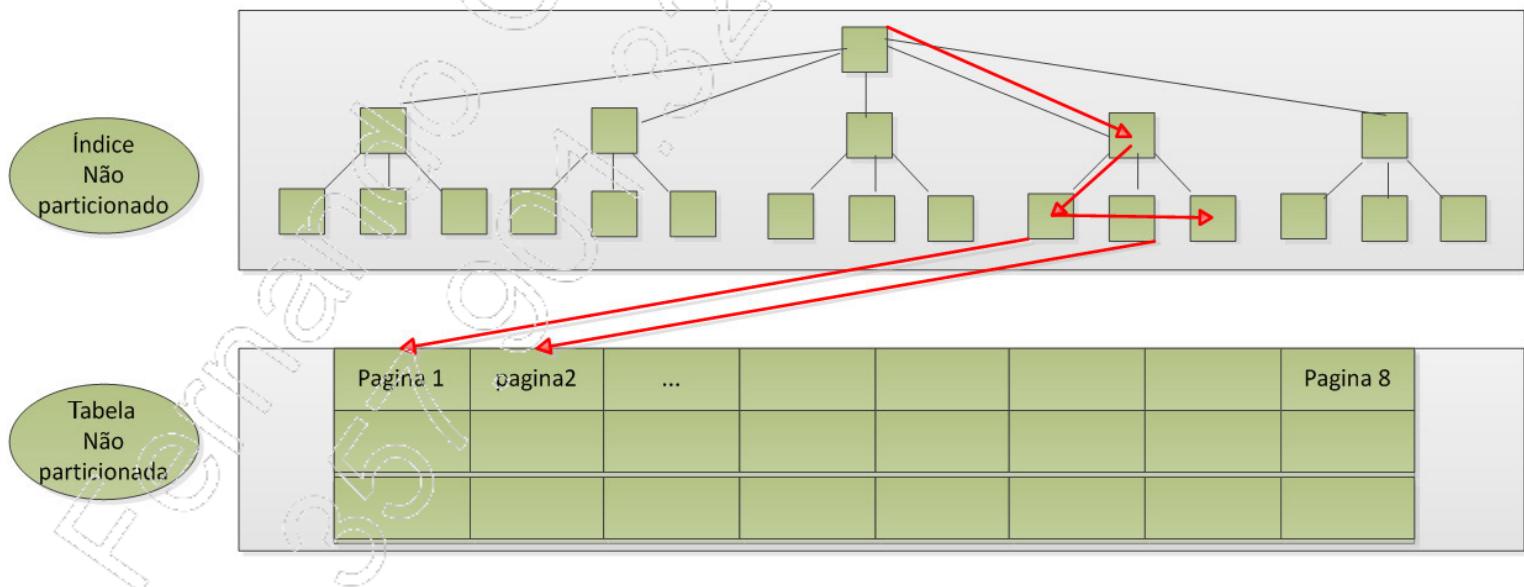
Tabelas particionadas têm a finalidade de dividir e separar os dados contidos em tabelas com grandes volumes de dados. Podemos falar em termos de milhões de registros quando o assunto é particionamento. A tabela particionada é dividida em tabelas menores chamadas de partições. Essas partições são criadas em função da seleção de colunas que oferecem valores para as chamadas funções de particionamento, as quais recebem os valores das colunas selecionadas e definem em qual partição a informação será armazenada. Existe uma grande quantidade de pontos positivos no particionamento:

- Melhor gerenciamento do espaço utilizado;
- Melhoria nos processos de consulta (desde que usem dados apenas da partição corrente);
- Movimentação de partições entre diferentes grupos de arquivos (FILEGROUPS);
- Cada partição pode ser truncada e movida individualmente.

Também existem aspectos negativos no particionamento de tabelas:

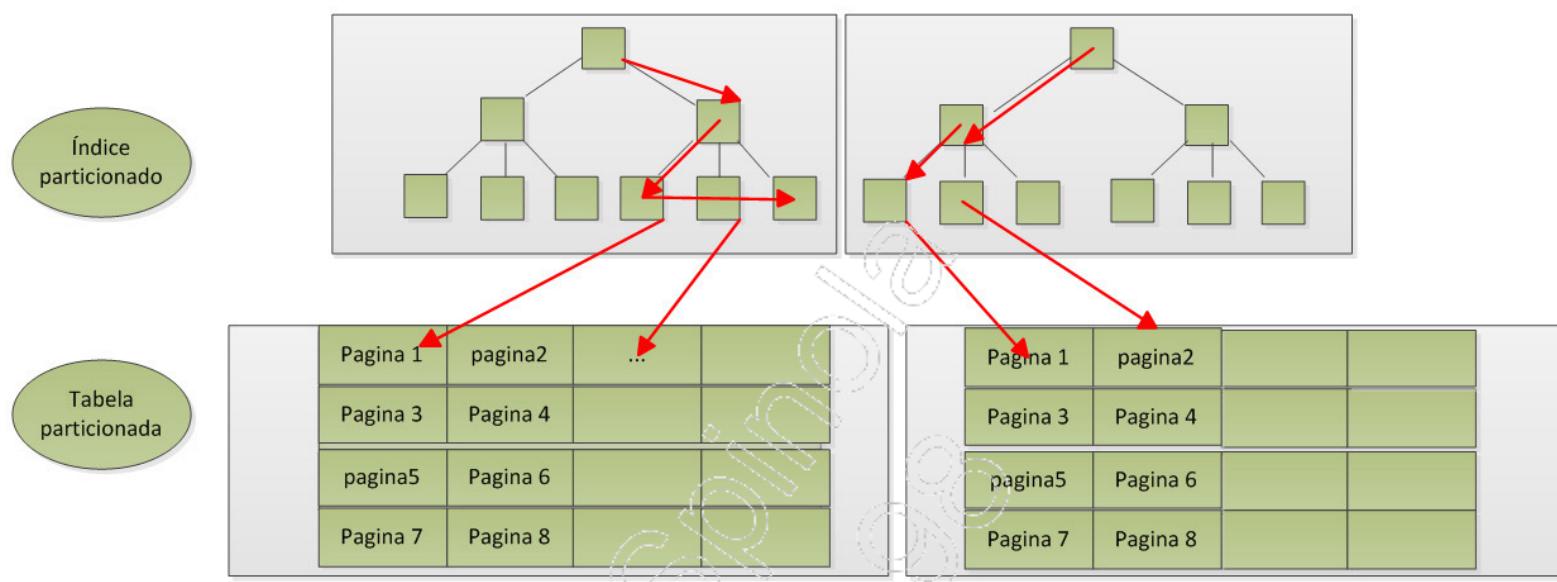
- Usar uma chave de particionamento não adequada pode levar a uma dispersão ruim dos dados, acarretando ações de leituras e de gravação (intrapartição). Um erro na escolha pode comprometer todo o processo de particionamento, implicando em perda de performance;
- É importante que a maioria das consultas use o mesmo critério do particionamento, do contrário ocorrerá intrapartição;
- A coluna que sofre o particionamento não deve ser alterada, pois isso implicaria eventualmente em movimentar o registro entre partições para acomodar no novo valor atualizado. Isso é dramático para o particionamento;
- Se os índices também não forem particionados (de preferência usando a mesma chave de partição), operações de truncate na partição irão invalidar o índice em questão.

Vejamos o diagrama de uma tabela e índices não particionados:



Numa situação normal, no caso de uma pesquisa indexada, o índice será percorrido do nível superior (root) até o menor nível (leaf). Nesse exemplo, o índice apresenta três níveis. Uma vez localizados os valores (índice não único), os dados serão localizados dentro dos respectivos extents e páginas. Esse exemplo utiliza índice não clusterizado.

Vejamos agora um diagrama de índices e tabelas particionadas:



Nesse caso, o tamanho de cada índice será menor e, consequentemente, as partições tornam as tabelas menores também, diminuindo o volume de leitura necessária para encontrar a informação. Esse exemplo utiliza índice não clusterizado.

O particionamento deve ser iniciado primeiramente com a seleção das colunas que servirão de base para isso, podendo ser uma ou mais. Colunas com alto grau de particionalidade são as que envolvem datas, pois normalmente as pesquisas tendem a usar datas como critérios de seleção. Imaginemos, por exemplo, uma empresa que emite milhares de notas fiscais por dia. A data de emissão da nota seria uma boa forma de particionarmos, já que boa parte das pesquisas ocorre dentro do mesmo mês em que a nota foi emitida. Outra boa indicação é que a data da emissão não poderia sofrer update.

SQL 2014 - Módulo III

Vejamos agora como criar a função do particionamento:

```
CREATE PARTITION FUNCTION FUN_PART_NF (DATETIME)
AS RANGE RIGHT FOR VALUES ('2011/01/01','2011/02/01','2011/03/01',
,'2011/04/01',
,'2011/05/01','2011/06/01','2011/07/01','2011/08/01',
,'2011/09/01','2011/10/01','2011/11/01','2011/12/01',
,'2014/01/01','2014/02/01','2014/03/01','2014/04/01',
,'2014/05/01','2014/06/01','2014/07/01','2014/08/01',
,'2014/09/01','2014/10/01','2014/11/01','2014/12/01',
,'2013/01/01','2013/02/01','2013/03/01','2013/04/01',
,'2013/05/01','2013/06/01','2013/07/01','2013/08/01',
,'2013/09/01','2013/10/01','2013/11/01','2013/12/01',
,'2014/01/01','2014/02/01','2014/03/01','2014/04/01',
,'2014/05/01','2014/06/01','2014/07/01','2014/08/01',
,'2014/09/01','2014/10/01','2014/11/01','2014/12/01');
```

Vamos criar o schema da partição vinculando a função de particionamento e armazenando-a no grupo de arquivo (FILEGROUP) chamado **DADOS**:

```
CREATE PARTITION SCHEME PART_TO_DADOS
AS PARTITION FUN_PART_NF
ALL TO ((DADOS01,DADOS02))
```

Agora vamos criar a tabela apontando para o schema de particionamento:

```
CREATE TABLE NOTA_FISCAL (
NUM_NF      INT,
NUM_SERIE   CHAR(2),
DAT_EMISSAO DATETIME,
VAL_NF       DECIMAL(12,2),
COD_CLIENTE INT,
VAL_DESCONTO DECIMAL(12,2),
CONSTRAINT NF_PK PRIMARY KEY (NUM_NF,NUM_SERIE),
CONSTRAINT NF_VAL_CK CHECK (VAL_NF >0))
ON PART_TO_DADOS(DAT_EMISSAO);
```

3.2.8. Tabelas com compressão

A compressão de dados pode ocorrer tanto no nível de linha quanto no nível de página, tanto para tabelas quanto para índices. Tanto tabelas normais, quanto particionadas podem sofrer compressão. A compressão é interna, ou seja, no momento do armazenamento, o dado sofre a compressão necessária, ajudando assim a diminuir o tamanho do banco de dados e também ajudando a diminuir a necessidade de leituras e gravações em disco (I/O). Existe um problema, porém, que trata da compressão e descompressão. Elas necessitam de recursos ligados à CPU e isto pode exigir mais memória e processador para realizar estas operações.

A compressão pode ser aplicada aos seguintes objetos:

- Tabelas sem índices clusterizados;
- Tabelas com índices clusterizados;
- Índices não clusterizados;
- Visão indexada;
- Tabelas e índices particionados (compressão aplicada a cada partição individualmente).

A compressão não pode ser aplicada aos seguintes objetos:

- Tabelas de sistema (system tables);
- Visões de sistema;
- Tabelas cujo tamanho da linha ultrapasse 8060 bytes.

A compressão está disponível apenas na versão Enterprise do SQL Server 2014 e recomenda-se que campos de tamanhos variáveis como **varchar** e **nvarchar** sejam alterados para **char** e **nchar**. Além disso, quando houver compressão, caso a linha tenha menos de 8060 bytes no momento da inserção, se um update tentar aumentá-la para mais de 8060 bytes, esta operação será bloqueada.

A compressão de página afeta todas as linhas que residem na página. Esse tipo de compressão tende a ser mais eficiente, pois controla o conjunto de dados presentes na página. A compressão de linha afeta cada linha individualmente, consumindo menos recursos para a sua realização.

3.2.9. Tabelas baseadas em arquivos (FileTable)

Tipo especial de tabela utilizada por aplicações que precisem realizar buscas de arquivos e diretórios em discos através do banco de dados. Essas tabelas usam uma API do sistema operacional e possuem as seguintes funcionalidades:

- Acesso de streaming não transacional e atualização para dados FILESTREAM;
- Suporte nativo e gerenciamento de diretórios e arquivos;
- Armazenamento de atributos de arquivos, tais como nome, data de criação e data de modificação;
- Suporte para APIs de gerenciamento de arquivos e diretórios do Windows.

As tabelas baseadas em arquivos servem para que o banco de dados trate de dados não estruturados, permitindo que esses dados possam ser trabalhados por servidores de arquivos para FileTables, para integrar esses servidores ao gerenciamento de dados estruturados.

Uma tabela baseada em arquivo (FILETABLE) representa uma hierarquia de diretórios e arquivos, armazenando todos os dados a todos os nós dessa hierarquia, tanto no nível de arquivos quanto no dos diretórios que os armazenam.

Cada registro de uma tabela baseada em arquivo (FILETABLE) representa um arquivo ou um diretório. Cada linha contém as seguintes colunas:

- Coluna **FILE_STREAM** contendo o nome do arquivo. Para diretórios, o valor é nulo;
- Coluna **STREAM_ID** contendo um identificador do tipo **GUID**;
- Coluna **PATH_LOCATOR** contendo o nome do arquivo e coluna **PARENT_PATH_LOCATOR** indicando o nome do diretório relacionado ao arquivo;
- Atributos de arquivo, no total de dez atributos;
- Coluna de texto que oferece suporte à pesquisa de texto completo e pesquisa semântica em documentos e arquivos.

Uma tabela baseada em arquivos possui gatilhos e restrições para manter o conteúdo dos arquivos.

Quando um banco de dados é configurado para acesso não transacional, a hierarquia do recurso FILETABLE é exposta no compartilhamento FILESTREAM, configurado no momento da instalação do SQL Server, oferecendo, por fim, o acesso para aplicativos do sistema operacional Windows.

A configuração das tabelas baseadas em arquivos (FILETABLE) é separada do recurso de FILESTREAM visto na instalação do SQL Server.

Não é possível realizar acesso não transacional dos dados contidos no FILESTREAM a não ser pelo recurso de tabelas baseadas em arquivos (FILETABLE). O recurso de acesso não transacional pode ser habilitado e desabilitado para cada banco de dados da instância.

SQL 2014 - Módulo III

Para criarmos uma tabela baseada em arquivo, usamos o comando **CREATE TABLE**. Vejamos a seguir a sintaxe para criação de uma FILETABLE:

```
CREATE TABLE nome_tabela AS FileTable  
[WITH (  
    [configurações], [configurações]  
) ]
```

Em que:

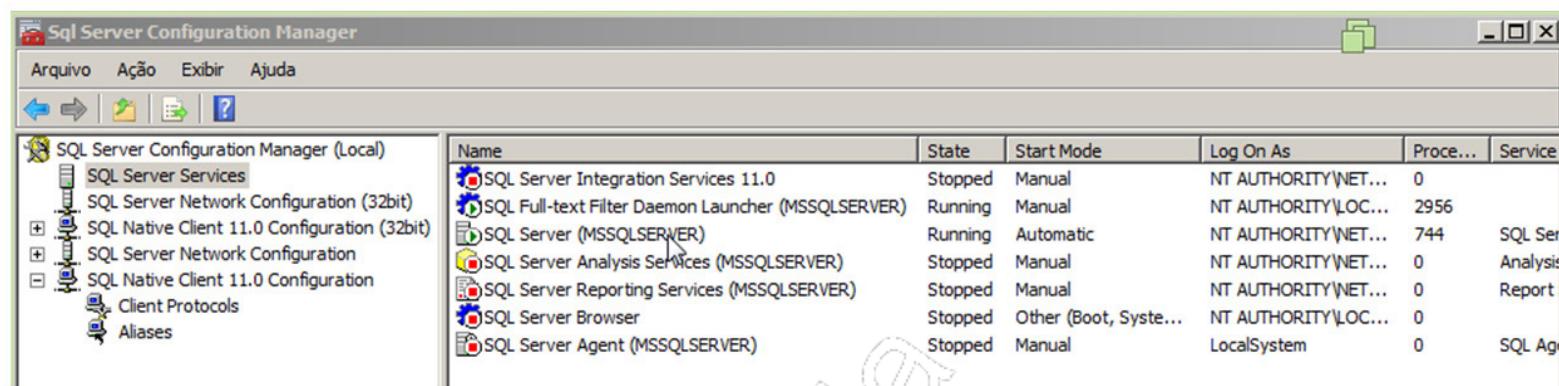
- **nome_tabela**: É o nome da tabela desejada;
- **Configurações**:
 - **FILETABLE_DIRECTORY**: Especifica o diretório que será raiz para todos os arquivos e diretórios armazenados em uma tabela baseada em arquivos (FILETABLE). Este nome deve ser exclusivo para todos os nomes de diretórios da tabela baseada em arquivo (FILETABLE). Não há distinção entre maiúsculas e minúsculas. Esse valor usa o tipo de dados **nvarchar(255)**. O nome deve obedecer aos requisitos do sistema operacional para um nome de arquivo, devendo ser exclusivo entre todos os nomes de diretórios da tabela baseada em arquivo. Se não for especificado um nome de diretório ao criar uma filetable, o nome desta última será usado como nome do diretório;
 - **FILETABLE_COLLATE_FILENAME**: Especifica o nome do collation para ser aplicado ao nome da coluna da tabela baseada em arquivo (FILETABLE). Se o valor para esta configuração não for definido, a coluna herdará o collation do banco de dados corrente.

Exemplo:

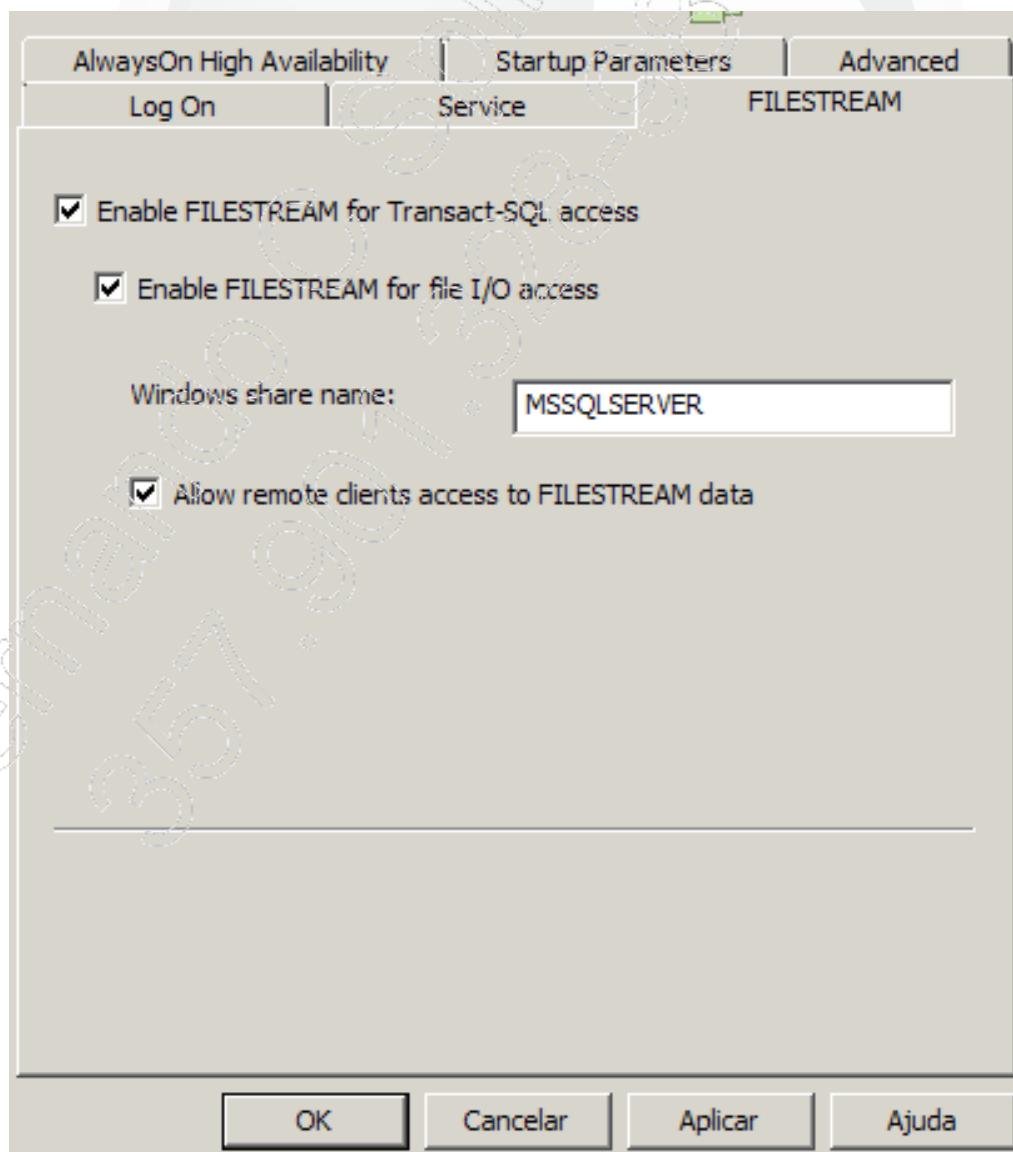
```
CREATE TABLE DOCUMENTOIMPACTA AS FILETABLE  
WITH (  FileTable_Directory = 'doc_impacta',  
        FileTable_Collate_FileName = database_default  
    );  
GO;
```

Gerenciando tabelas e outros objetos

Para configurarmos o recurso de FILETABLE, devemos habilitar o recurso de FILESTREAM. Para isso, vamos acessar a ferramenta de configuração do SQL Server. Abra a pasta **SQL Server Services**, localize a instância que deseja habilitar, pressione o botão direito do mouse e selecione a opção **Propriedades**:



A tela a seguir será exibida:



Selecione a opção Habilitar FILESTREAM para acesso Transact-SQL (**Enable FILESTREAM for Transact-SQL access**). Caso queira ler e gravar em arquivos, selecione a opção que habilita FILESTREAM para acesso de leitura e gravação de arquivos (**Enable FILESTREAM for file I/O access**). Se quiser que conexões remotas possam acessar o FILESTREAM, habilite a última opção da tela.

Execute o comando a seguir e reinicialize o serviço de instância:

```
exec sp_configure filestream_access_level,2  
RECONFIGURE
```

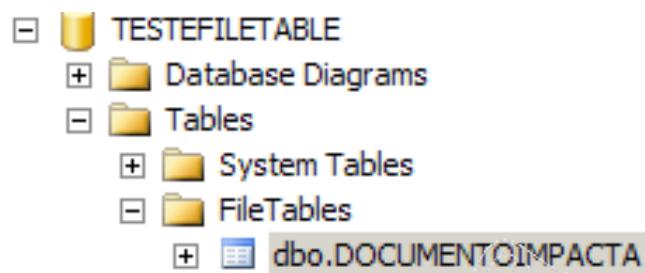
- Criando um banco de dados para uso com o filestream:

```
create database TESTEFILETABLE  
ON PRIMARY  
(  
    NAME=N'TESTEFILETABLE_DADOS',  
    FILENAME = N'c:\app\TESTEFILETABLE.MDF'  
)  
FILEGROUP FILESTREAMImpacta CONTAINS FILESTREAM  
(  
    NAME=N'TESTEFILETABLE',  
    FILENAME = N'c:\teste'  
)  
LOG ON  
(  
    NAME=N'TESTEFILETABLE_LOG',  
    FILENAME = N'c:\app\TESTEFILETABLE_LOG.LDF'  
)  
WITH FILESTREAM  
(  
    NON_TRANSACTED_ACCESS=FULL,  
    DIRECTORY_NAME=N'FileTableImpacta'  
)
```

- Criando uma tabela baseada em arquivo:

```
CREATE TABLE DOCUMENTOIMPACTA AS FILETABLE  
WITH ( FileTable_Directory = 'doc_impacta'  
);
```

A seguir, veremos a tabela criada:



- Considerações:
 - Tabelas normais não podem ser convertidas em tabelas baseadas em arquivo;
 - Não é possível criar uma constraint como parte de um comando de criação de tabelas baseadas em arquivo;
 - Não é possível criar uma tabela do tipo FILETABLE como tabelas temporárias;
 - Não é possível criar uma tabela nos bancos de dados do sistema.
- Alterando uma tabela baseada em arquivo:

```
ALTER TABLE nome_tabela  
SET (  
    [configurações], [configurações]  
)
```

- Não é possível alterar o valor da configuração **FILETABLE_COLLATE_FILENAME**;
- Não é possível alterar, remover ou desabilitar as colunas definidas para um FILEGROUP;
- Não é possível adicionar novas colunas.

- Para eliminar uma tabela baseada em arquivo:

```
DROP TABLE nome_tabela
```

- Verificando todos os objetos para todas as FileTables:

```
SELECT * FROM SYS.FILETABLE_SYSTEM_DEFINED_OBJECTS;
```

Ou:

```
SELECT OBJECT_NAME(PARENT_OBJECT_ID) AS 'FILETABLE', OBJECT_NAME  
(OBJECT_ID) AS 'System-Defined Object'  
FROM SYS.FILETABLE_SYSTEM_DEFINED_OBJECTS  
ORDER BY FILETABLE, 'System-Defined Object'
```

3.3. Visões (Views)

Visões são objetos de banco de dados que armazenam informações de forma lógica. Neste caso, os dados contidos em uma visão são armazenados em uma ou mais tabelas ou em outras visões.

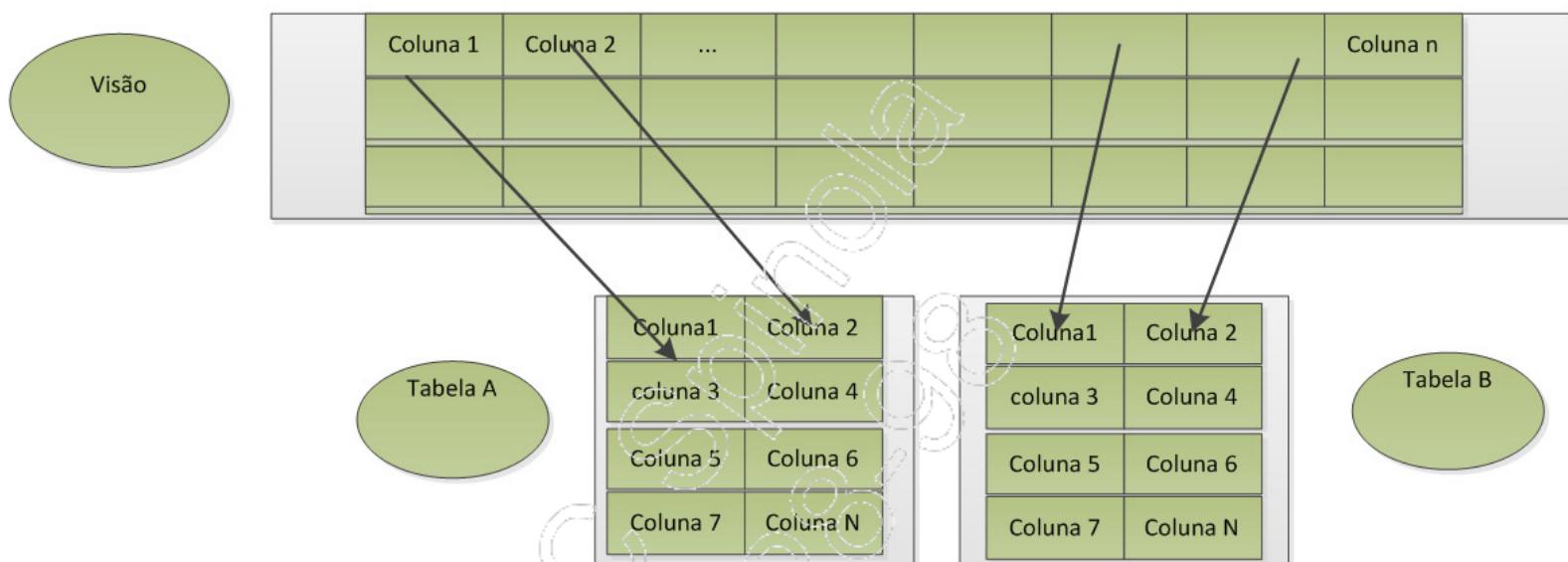
A visão permite que os dados possam ser agrupados conforme uma determinada necessidade. O resultado disso é um conjunto de dados semelhantes a uma tabela.

O uso de visão apresenta os seguintes aspectos positivos:

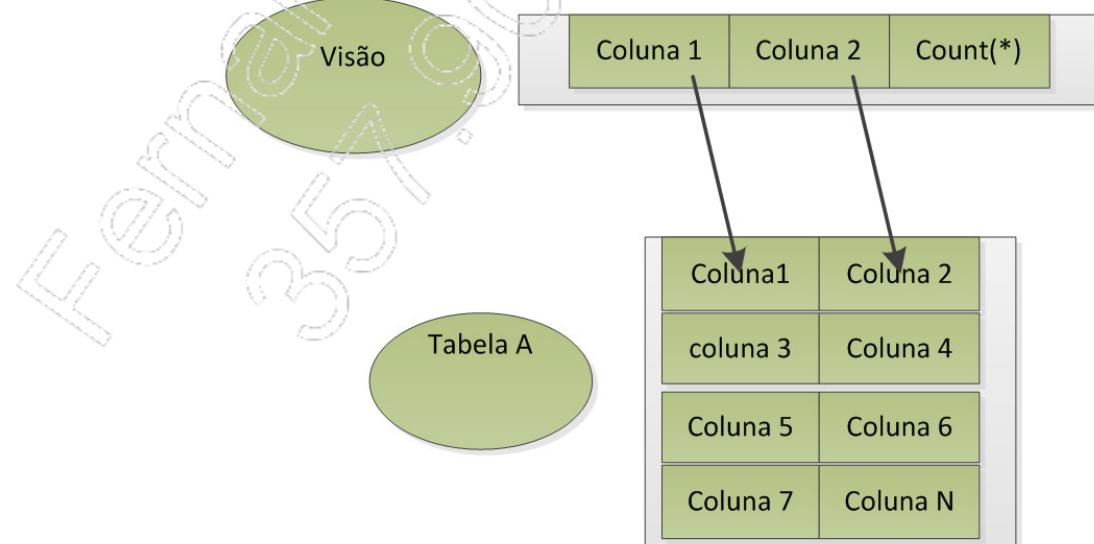
- Formatação de dados complexos, tirando a complexidade dos relacionamentos e da apresentação dos dados;
- Proteção no acesso aos dados, pois é possível exibir apenas dados (colunas) desejados(as), impedindo o acesso a dados privilegiados;
- Segurança contra manipulação indevida de dados (comandos **INSERT**, **UPDATE**, **DELETE** e **MERGE**).

O que podemos caracterizar como aspecto negativo é que a operação de consulta tende a ser menos eficiente do que se fosse realizada contra a própria tabela. Isso ocorre porque o comando realizado contra a visão deverá ser refeito para que atinja as tabelas, o que envolve um processamento adicional.

Vejamos a representação de uma visão englobando duas ou mais tabelas:



Vejamos a representação de uma visão englobando uma única tabela:



SQL 2014 - Módulo III

Operações de Insert podem ser aplicadas a visões caso estas condições sejam satisfeitas:

- Todas as colunas **not null** (sem cláusula DEFAULT) devem estar presentes;
- As colunas **not null** que não estiverem presentes na visão devem ter cláusula DEFAULT;
- Não utilizem funções nem agrupamento;
- Não utilizem expressões aritméticas;
- Utilizem apenas uma tabela.

Operações de update podem ser aplicadas a visões caso estas condições sejam satisfeitas:

- Não utilizem funções nem agrupamento;
- Utilizem apenas uma tabela.

Operações de delete podem ser aplicadas a visões caso estas condições sejam satisfeitas:

- Não utilizem funções nem agrupamento;
- Utilizem apenas uma tabela.

3.3.1. Criação de uma visão

A seguir, mostraremos a sintaxe da visão:

```
CREATE VIEW [ schema_name . ] view_name [ (column [ ,...n ] ) ]
[ WITH <view_attribute> [ ,...n ] ]
AS select_statement
[ WITH CHECK OPTION ] [ ; ]

<view_attribute> ::==
{
    [ ENCRYPTION ]
    [ SCHEMABINDING ]
    [ VIEW_METADATA ] }
```

Em que:

- **ENCRYPTION**: Realiza a encriptação do código-fonte impedindo que este possa ser visualizado;
- **SCHEMABINDING**: Impede a alteração das estruturas que compõem as tabelas;
- **VIEW_METADATA**: Quando especificada a opção **VIEW_METADATA**, o SQL envia somente as informações de exibição da consulta da View, não disponibilizando as informações dos metadados das tabelas bases.

Exemplo de criação de visão encriptando o código-fonte:

```
CREATE VIEW V_ALUNO_MATRICULA
AS SELECT A.* , B* FROM MATRICULA. A JOIN ALUNO B
ON A.COD_ALUNO =B.COD_ALUNO
WITH ENCRYPTION
```

Exemplo de criação de visão que impede a alteração das tabelas originais:

```
CREATE VIEW V_ALUNO_MATRICULA
AS SELECT A.* , B* FROM MATRICULA. A JOIN ALUNO B
ON A.COD_ALUNO =B.COD_ALUNO
WITH SCHEMABINDING
```

Para encontrarmos as visões criadas, podemos utilizar o seguinte comando:

```
SELECT * FROM SYS.VIEWS;
```

3.3.2. Alterando a visão

A alteração é possível através do comando **ALTER**. Podemos alterar a visão adicionando novas colunas ou mesmo eliminando colunas, alterando condições, entre outras operações.

A seguir, temos a sintaxe para o comando **ALTER** em uma visão:

```
ALTER VIEW [ schema_name . ] view_name [ ( column [ ,...n ] ) ]
[ WITH <view_attribute> [ ,...n ] ]
AS select_statement
[ WITH CHECK OPTION ] [ ; ]

<view_attribute> ::=

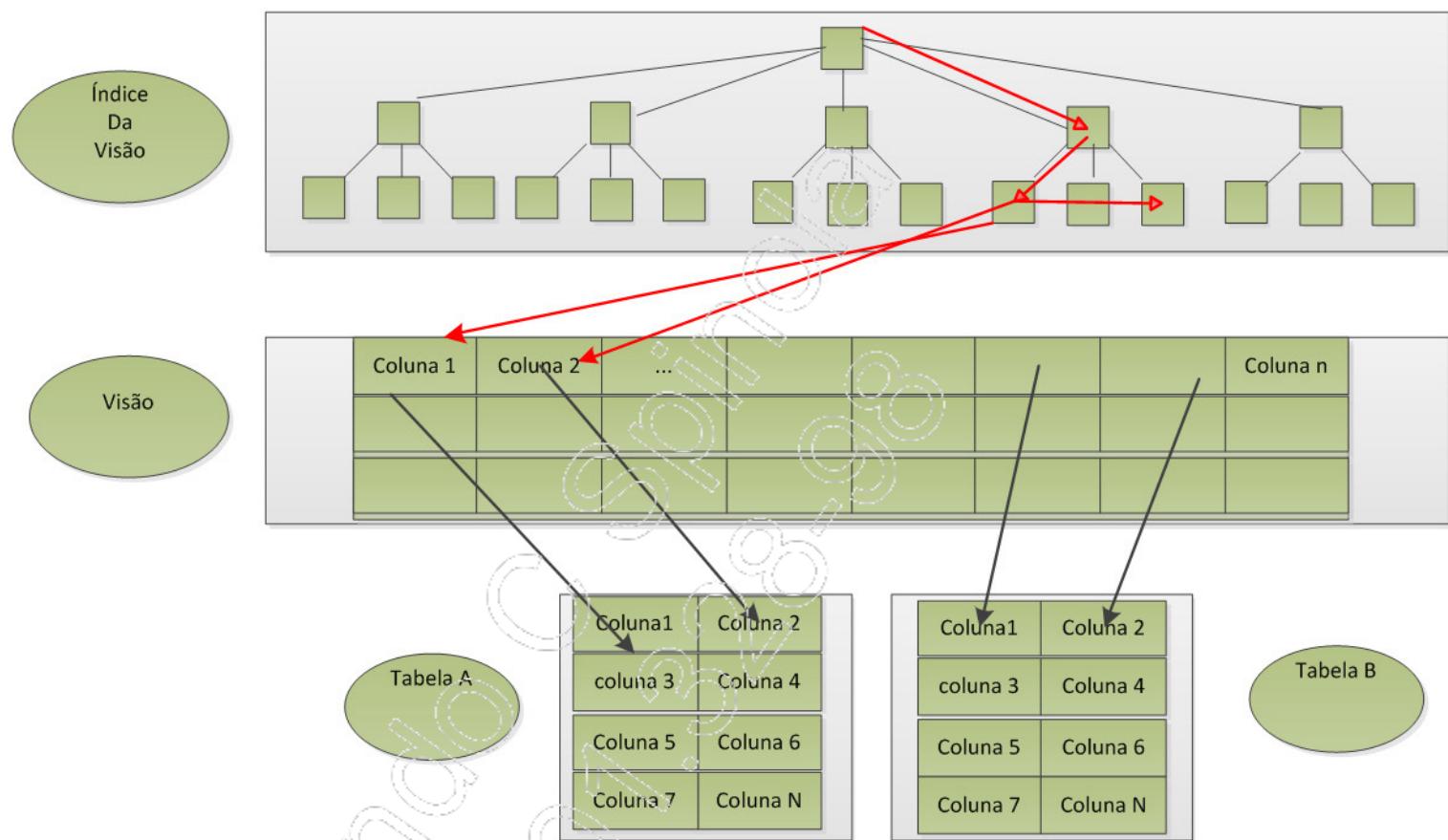
{
    [ ENCRYPTION ]
    [ SCHEMABINDING ]
    [ VIEW_METADATA ]
```

Exemplo de alteração de uma visão:

```
ALTER VIEW V_ALUNO_MATRICULA
AS SELECT A.COD_MATRICULA,A.DAT_MATRICULA,A.COD_ALUNO,A.NOM_ALUNO
FROM MATRICULA A JOIN ALUNO B
ON A.COD_ALUNO =B.COD_ALUNO
WITH SCHEMABINDING
```

3.3.3. Visões indexadas

Visões podem ser indexadas para melhorar a performance no acesso aos dados nas tabelas que constituem a base das visões. A seguir, demonstramos o esquema que ilustra uma visão indexada:



Exemplo de visão indexada:

```
CREATE VIEW V_ALUNO AS  
SELECT COD_ALUNO, NOM_ALUNO  
FROM DBO.ALUNO;  
  
CREATE UNIQUE CLUSTERED INDEX ALUNO_UCI ON DBO.V_ALUNO(COD_ALU-  
NO);
```

3.4. Criando sequências

No SQL Server 2014, é possível criarmos um objeto chamado **Sequência**. Esse objeto visa retornar um número sequencial, que pode ser utilizado em qualquer aplicação, inclusive como base para a criação de chaves primárias em registros.

A forma como a sequência é controlada depende de aplicação. Nada impede que duas ou mais aplicações utilizem a mesma sequência para obter valores para inserção de dados em uma única tabela, ou mesmo para utilização em diferentes tabelas. Apenas a aplicação pode impedir ou vincular o uso de uma sequência.

Não se recomenda o uso de sequências para valores que não possam conter “furos”, pois, entre o momento em que se captura o valor da sequência e o momento em que esta é efetivamente utilizada, pode ocorrer algum tipo de falha ou mesmo a aplicação sofrer um comando rollback.

Vejamos a sintaxe para criação de sequências:

```
CREATE SEQUENCE [schema_name .] sequence_name
    [ AS [ built_in_integer_type | user-defined_integer_type ] ]
    [ START WITH <constant> ]
    [ INCREMENT BY <constant> ]
    [ { MINVALUE [ <constant> ] } | { NO MINVALUE } ]
    [ { MAXVALUE [ <constant> ] } | { NO MAXVALUE } ]
    [ CYCLE | { NO CYCLE } ]
    [ { CACHE [ <constant> ] } | { NO CACHE } ]
    [ ; ]
```

Pode-se definir o valor inicial da sequência, como ela será incrementada, se terá um limite ou não e se, ao atingir um limite, a sequência deve ser reciclada ou não reciclada. Ainda para melhorar a performance da sequência, pode-se definir que ela seja gerenciada em memória (CACHE).

Exemplo de criação de sequências:

```
CREATE SEQUENCE SEQ_ALUNO;
```

No exemplo anterior, a sequência iniciará em 1, será incrementada de 1 em 1, não terá limite e não sofrerá cache em memória:

```
CREATE SEQUENCE SEQ_ALUNO
START WITH 1000
INCREMENT BY 10
MINVALUE 10
MAXVALUE 10000
CYCLE CACHE 10;
```

No exemplo anterior, a sequência inicia no número 1000, é incrementada de 10 em 10 e termina em 10000. Ao final, será retornada ao valor original 10 (**MINVALUE**).

Para utilizarmos o valor de uma sequência em uma inserção, devemos usar a cláusula **NEXT VALUE FOR** e o nome da sequência.

```
INSERT INTO ALUNO (COD_ALUNO, NOM_ALUNO)
VALUES (NEXT VALUE FOR DBO.SEQ_ALUNO, 'TESTE');
```

Para encontrarmos as sequências criadas, podemos utilizar o seguinte comando:

```
SELECT * FROM SYS.SEQUENCES;
```

3.5. Criando sinônimos

Sinônimos são recursos que permitem o mascaramento do schema de um objeto, por exemplo, uma tabela. Todos os objetos pertencem a um schema nomeado ou, caso não existam schemas criados, podemos utilizar o schema DBO. À medida que aumentam a complexidade do modelo e a quantidade de objetos, podemos usar o recurso de criação de sinônimos.

Vejamos a seguir a sintaxe para criação de sinônimos:

```
CREATE SYNONYM [ schema_name_1. ] synonym_name FOR <object>

<object> ::= =
{
    [ server_name.[ database_name ] . [ schema_name_2 ].|
    database_name . [ schema_name_2 ].| schema_name_2. ] object_name
}
```

Criando um sinônimo para uma tabela:

```
CREATE SYNONYM TAB_ALUNO FOR DBO.ALUNO;

SELECT * FROM TAB_ALUNO;
```

Acessando um sinônimo em uma função:

```
CREATE FUNCTION dbo.Fun_teste (@valor int)
RETURNS int
WITH EXECUTE AS CALLER
AS
BEGIN
IF @valor % 12 <> 0
BEGIN
    SET @valor += 12 - (@valor % 12)
END
RETURN(@valor);
END;
GO
CREATE SYNONYM FUN_TESTE_EXEMPLO FOR DBO.FUN_TESTE;
```

Pode-se criar sinônimos para tabelas, visões, procedimentos, funções, sequências, entre outros objetos.

3.6. Criando tipos

Tipos são objetos criados para definição dos chamados tipos de dados de usuários (USER TYPES). Isso permite que criemos um novo tipo de dado utilizando os existentes.

A seguir, temos a sintaxe para a criação de tipos:

```
CREATE TYPE [ schema_name. ] type_name
{
    FROM base_type
    [ ( precision [ , scale ] ) ]
    [ NULL | NOT NULL ]
    | EXTERNAL NAME assembly_name [ .class_name ]
    | AS TABLE ( { <column_definition> | <computed_column_definition> }
        [ <table_constraint> ] [ ,...n ] )
} [ ; ]

<column_definition> ::=

column_name <data_type>
    [ COLLATE collation_name ]
    [ NULL | NOT NULL ]
    [
        DEFAULT constant_expression ]
    | [ IDENTITY [ ( seed ,increment ) ] ]
]
[ ROWGUIDCOL ] [ <column_constraint> [ ...n ] ]

<data_type> ::=
[ type_schema_name . ] type_name
    [ ( precision [ , scale ] | max |
        [ { CONTENT | DOCUMENT } ] xml_schema_collection ) ]

<column_constraint> ::=
{ { PRIMARY KEY | UNIQUE }
    [ CLUSTERED | NONCLUSTERED ]
    [
        WITH ( <index_option> [ ,...n ] )
    ]
| CHECK ( logical_expression )
}
```

SQL 2014 - Módulo III

```
<computed_column_definition> ::=  
column_name AS computed_column_expression  
[ PERSISTED [ NOT NULL ] ]  
[  
{ PRIMARY KEY | UNIQUE }  
[ CLUSTERED | NONCLUSTERED ]  
[  
    WITH ( <index_option> [ ,...n ] )  
]  
| CHECK ( logical_expression )  
]  
  
<table_constraint> ::=  
{  
{ PRIMARY KEY | UNIQUE }  
[ CLUSTERED | NONCLUSTERED ]  
    ( column [ ASC | DESC ] [ ,...n ] )  
[  
    WITH ( <index_option> [ ,...n ] )  
]  
| CHECK ( logical_expression )  
}  
  
<index_option> ::=  
{  
    IGNORE_DUP_KEY = { ON | OFF }  
}
```

Vejamos um exemplo de criação de tipo:

```
CREATE TYPE ENDERECO AS TABLE  
(DES_LOGRADOURO    VARCHAR(50),  
NUM_LOGRADOURO    VARCHAR(10),  
DES_BAIRRO        VARCHAR(30),  
DES_CIDADE         VARCHAR(30),  
NUM_CEP            NUMBER(8));
```

Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo.

- O particionamento de tabelas auxilia na performance do acesso aos dados, desde que seja dentro da mesma partição. Se as operações ocorrerem fora da partição, elas podem provocar uma perda de performance ainda maior. Logo, a chave para o sucesso do particionamento é a escolha correta da coluna ou colunas que deverão ser particionadas;
- O uso excessivo de tabelas temporárias pode provocar problemas de contenção e de aumento do banco de dados **tempdb**;
- Visões permitem ampliar a segurança de dados à medida que podem restringir colunas e linhas de tabelas, evitando assim que dados sensíveis possam ser acessados;
- Sequências permitem que possamos atribuir valores sequenciais a uma ou mais chaves primárias sem a necessidade de criação de lógica de aplicação para isso;
- Sinônimo é um recurso que permite o encapsulamento do schema dos objetos. Dessa forma, é possível criar objetos em schemas diferentes e uma mesma lógica de aplicação pode encontrar esses objetos, permitindo uma maior flexibilidade em sua gestão.

Gerenciando tabelas e outros objetos

Teste seus conhecimentos

3

Fernando Gómez
357.907.3000



IMPACTA
EDITORA

1. Qual das alternativas a seguir não é um tipo de constraint?

- a) Primary Key
- b) Foreign Key
- c) Check
- d) Identity
- e) Not Null

2. Qual a vantagem da utilização dos sinônimos?

- a) Não devem ser utilizados.
- b) Permite que as views possam ser criadas em qualquer schema.
- c) Permite que as tabelas possam ser criadas em qualquer schema.
- d) Permite acesso direto na tabela.
- e) Permite o mascaramento do schema de objetos.

3. Por que a segurança de dados pode ser implementada com o uso de visões?

- a) Impede o acesso aos dados diretamente na tabela.
- b) Bloqueia o acesso na tabela.
- c) Cria uma regra para bloqueio de acesso.
- d) Não há diferença no tipo de acesso aos dados.
- e) Melhora a performance, mas não o acesso.

4. O particionamento de tabelas é recomendado em quais situações?

- a) Para todas as tabelas.
- b) Somente para tabelas de sistema.
- c) Para tabelas que possuam relacionamentos.
- d) Para tabelas com grande volume de dados.
- e) Para qualquer sistema.

5. Qual a vantagem da utilização de FileTable?

- a) Os arquivos binários não são armazenados no banco e sim em diretórios.
- b) O SQL suporta tipos binários e não há vantagem em relação ao FileTable.
- c) A vantagem está na performance.
- d) Todo armazenamento deve ser realizado em tabelas.
- e) Não é um recurso seguro.

Gerenciando tabelas e outros objetos

Mãos à obra!

3

*Fernando
357.907.*



IMPACTA
EDITORA

Laboratório 1

A – Criando um particionamento para tabelas de um banco de dados

Scripts utilizados neste laboratório:

A pasta **Capitulo_03** contém o **Script_01** com os comandos necessários para a realização deste laboratório. Utilize os scripts se não desejar digitar os respectivos comandos ou apenas para corrigir sua execução.

1. Crie um banco de dados com 3 FileGroups além do PRIMARY:

Nome do Database: TESTE_PARTICAO				
Collation: Default				
Script: C:\SQLServer2014 - Módulo III\Capítulo_03\Script_01.sql				
Arquivo(s) de Dado(s)				
Filegroup: Primary				
Nome Lógico	Nome Físico	Tam. Inicial	Tam. Máximo	Crescimento
TESTE_PARTICAO_PRIM	C:\DADOS\TESTE_PARTICAO\TESTE_PARTICAO_PRIM.MDF	10MB	10MB	5MB
Filegroup: FG1				
TESTE_PARTICAO_FG1	C:\DADOS\TESTE_PARTICAO\TESTE_PARTICAO_FG1.NDF	10MB	10MB	5MB
Filegroup: FG2				
TESTE_PARTICAO_FG2	C:\DADOS\TESTE_PARTICAO\TESTE_PARTICAO_FG2.NDF	10MB	10MB	5MB
Filegroup: FG3				
TESTE_PARTICAO_FG3	C:\DADOS\TESTE_PARTICAO\TESTE_PARTICAO_FG3.NDF	10MB	10MB	5MB
Arquivo(s) de Log(s)				
TESTE_PARTICAO_LOG	C:\DADOS\TESTE_PARTICAO\TESTE_PARTICAO_LOG.LDF	10MB	10MB	5MB

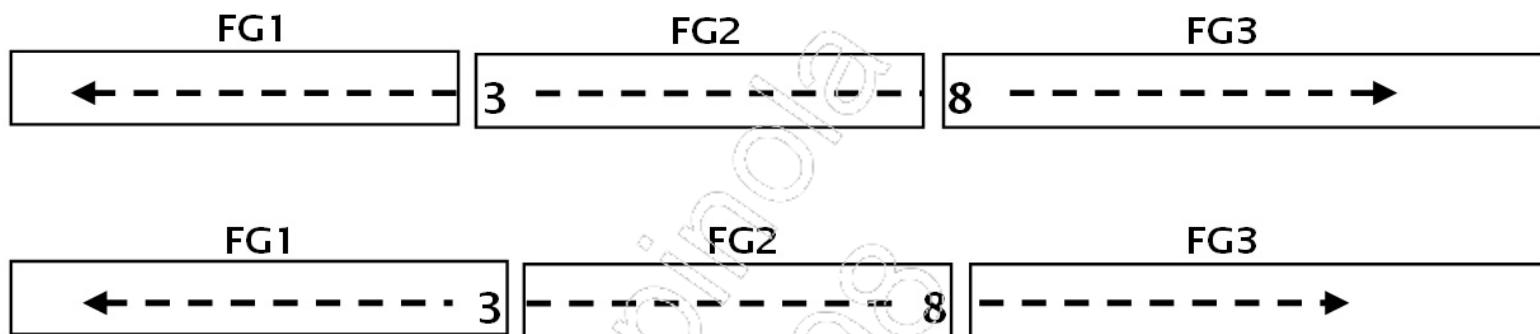
2. Coloque o banco TESTE_PARTICAO em uso;

SQL 2014 - Módulo III

3. Crie uma função de particionamento conforme a regra a seguir:

COD_DEPTO do funcionário	Nome do filegroup
Menor que 3	FG1
De 3 até 7	FG2
De 8 em diante	FG3

4. Crie um esquema de partição que associe cada partição (grupo de linhas da tabela) a um filegroup:



5. Crie a tabela **EMPREGADOS** e associe-a ao schema de particionamento criado no passo anterior:

```
CODFUN int IDENTITY          NOT NULL  
NOME      varchar(35)        NOT NULL,  
COD_DEPTO int                NOT NULL  
COD_CARGO int  
DATA_ADMISSAO datetime  
SALARIO   numeric(18,2)
```

6. Execute a consulta para verificar a definição do particionamento;

7. Insira 2 linhas na tabela **EMPREGADOS** com **COD_DEPTO < 3**;

8. Insira 5 linhas na tabela **EMPREGADOS** com **COD_DEPTO >= 3 AND COD_DEPTO < 8**;

9. Insira 2 linhas na tabela **EMPREGADOS** com **COD_DEPTO >= 8**;

10. Execute uma consulta e verifique os registros por partição.

Laboratório 2

A - Criando um objeto SEQUENCE e utilizando-o na tabela DEPARTAMENTO

1. Crie uma tabela **DEPARTAMENTO** com os seguintes campos:

```
COD_DEPTO      INT PRIMARY KEY,  
NOME          VARCHAR(10)
```

2. Crie uma SEQUENCE de nome **SEQ_DEPTO** com início em 1;
3. Insira dois departamentos, **Contabilidade** e **TI**, na tabela **DEPARTAMENTO**, utilizando a SEQUENCE criada;
4. Consulte a SEQUENCE existente no banco.

Índices

4

- ✓ Índices;
- ✓ Determinando a criação de um índice;
- ✓ Obtendo informações sobre os índices;
- ✓ Manutenção de índices;
- ✓ O otimizador e o plano de execução;
- ✓ Índices Full-Text.



IMPACTA
EDITORA

4.1. Introdução

Informações específicas de uma tabela ou de uma view podem ser encontradas de maneira mais rápida através dos índices. Por meio de chaves baseadas em colunas, os índices permitem definir a localização de uma determinada informação. Os índices otimizam não só a performance das queries, como também a leitura de informações encontradas em um banco de dados como um todo.

4.2. Índices

Com a utilização dos índices, podemos encontrar rapidamente uma determinada informação. O local em que essa informação está armazenada é indicado através de apontadores. Um índice também possui chaves definidas com base em uma ou mais colunas de uma tabela. Além de facilitar a busca de dados, os índices são capazes de suportar o uso de queries, além de diminuir o volume de dados que será lido para o retorno de resultados de uma query.

Os índices garantem, ainda, a integridade dos dados encontrados nas linhas de uma tabela.

Os índices podem ser adicionados ou retirados de uma tabela quando necessário sem comprometer o esquema do banco de dados, uma vez que eles não fazem parte do projeto lógico da tabela.

4.2.1. Estruturas de índices do SQL Server

Os índices são formados por várias páginas que, juntas, compõem uma estrutura similar a uma árvore, chamada de Árvore-B+ ou B-Tree. Nesta árvore, todos os caminhos que partem da “raiz” até as “folhas” possuem comprimentos iguais.

Um índice Árvore-B apresenta diversos níveis, sendo que cada um pode ser chamado de nível não-folha (non-leaf level) ou nível folha (leaf level). Enquanto os índices são armazenados em páginas de índices, os dados encontram-se nas páginas de dados.

Tanto em uma página de dados quanto em uma página de índices, podemos armazenar 8.096 bytes, pois cada uma possui 8.192 de tamanho (8K), sendo que o cabeçalho ocupa 96 bytes.

Nessa estrutura de árvore, encontramos as chaves construídas com base em uma ou várias colunas de uma view ou de uma tabela. Os valores chave são utilizados como referência pelo SQL Server para a busca de determinadas linhas de dados.

Duas estruturas diferentes de índices podem ser utilizadas, como podemos observar a seguir:

- **Clustered:** Neste tipo de estrutura, a tabela é ordenada fisicamente. Por meio do Clustered, é possível otimizar a performance de leituras baseadas na filtragem de dados de acordo com um alcance (range) de valores. É permitido o uso de apenas um índice Clustered por tabela, já que as linhas de dados só podem ser ordenadas de uma maneira;

Uma tabela que possui um índice Clustered é chamada de tabela Clustered, pois suas linhas passam a ser organizadas de forma ordenada. Caso a tabela não possua esse tipo de índice, a estrutura de organização dos dados será chamada de heap (pilha), pois se trata de um modo sem ordenação.

- **NonClustered:** Neste tipo de estrutura, os dados de uma tabela são ordenados de maneira lógica. Os índices NonClustered possuem valores chave, sendo que um ponteiro para a linha de dados é encontrado em cada entrada desses valores. Esse ponteiro é conhecido como **localizador de linha** ou **row locator**. Caso as páginas de dados estejam armazenadas em uma tabela Clustered, este localizador será a chave do índice Clustered. Mas se o localizador estiver em uma pilha, ele será um apontador da linha.

O uso do índice NonClustered é recomendado para leituras que permitem o retorno de uma quantidade pequena de dados de uma tabela. Com isso, podemos utilizar até 249 índices NonClustered por tabela. Para que algumas queries possam ser executadas por completo, devemos ampliar os limites da chave de índice existente. No SQL Server 2014, podemos adicionar colunas que não são definidas como chave no nível folha do índice NonClustered para que esse limite seja ultrapassado.

4.2.2. Índice Clustered

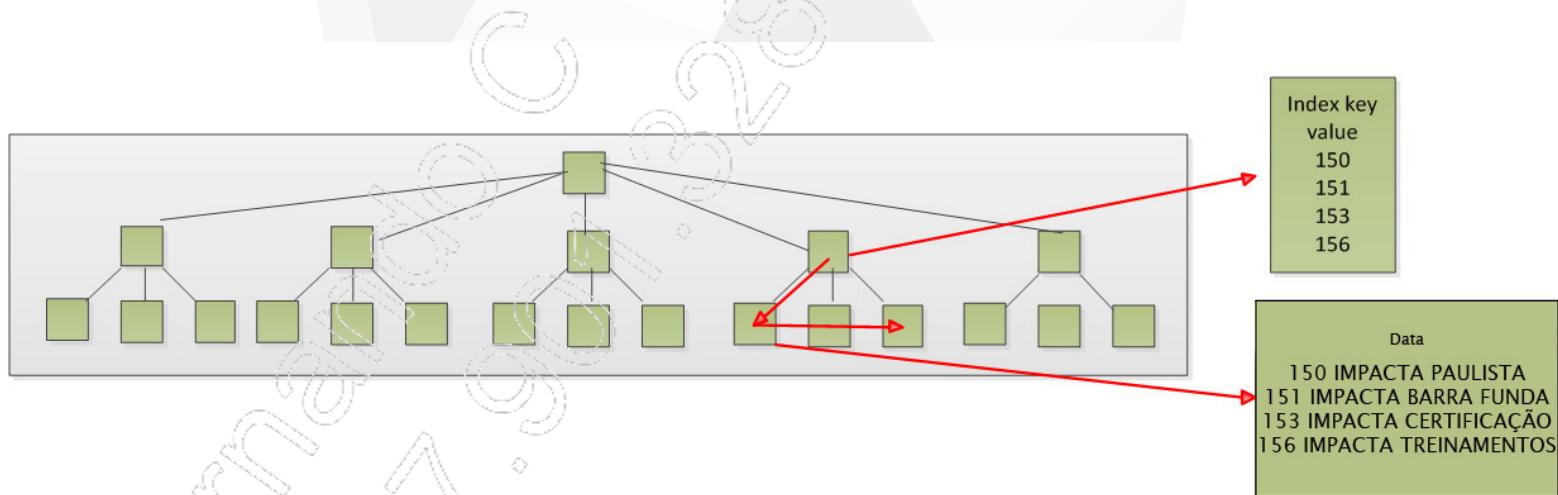
Os dados são ordenados fisicamente e armazenados na própria tabela pelo SQL Server com base nos valores de chave definidos pelo índice Clustered. As páginas de dados das tabelas são encontradas no nível folha da estrutura desse tipo de índice.

Os índices **Clustered** sempre devem ser criados antes dos índices **NonClustered**, pois eles são capazes de alterar a ordem física de linhas de dados de uma tabela. Esse procedimento também evita a reconstrução dos índices **NonClustered**.

O exemplo a seguir mostra como criar um índice **Clustered**:

```
CREATE CLUSTERED INDEX I_Funcionario_1  
ON Funcionario(Nome_Func)
```

Representação gráfica para índices clusterizados:



Para que o índice **Clustered** seja criado, o SQL será obrigado a reconstruir novamente a tabela inteira. Por essa razão, o banco de dados deve contar com um espaço livre de, pelo menos, 120% do tamanho da tabela que já contém dados. No entanto, essa quantidade é temporária, uma vez que o índice, depois de criado, consumirá apenas 5% a mais do espaço que já era ocupado antes pela tabela. O espaço ocupado pelo índice **Clustered** é menor quando comparado ao **NonClustered**, já que ele é capaz de ordenar a sua própria tabela.

Além de propiciar maior integração dos dados, o índice **Clustered** ainda permite a aplicação das queries utilizadas com maior frequência. Vejamos alguns casos específicos de uso desse índice com algumas constraints e propriedades:

- Por padrão, um índice **Clustered** sempre é criado em uma ou mais colunas quando geramos uma constraint **PRIMARY KEY**;
- Caso a propriedade **UNIQUE** não seja utilizada no momento em que o índice **Clustered** for criado, uma coluna uniqueifier de 4 bytes será acrescentada na tabela pelo Database Engine. Um valor torna único cada valor de chave referente a uma linha de dados, sendo que outros usuários não conseguem acessar esse tipo de valor;
- Um índice **Clustered**, bem como a definição de uma **PRIMARY KEY**, pode otimizar a performance das queries utilizadas para a busca de informações únicas encontradas em uma coluna (como o número de CPF dos funcionários). A combinação de colunas relacionadas (como rua e número) também deve ser considerada na utilização desse índice;
- Os índices **Clustered** também são indicados quando utilizamos queries que especificam uma busca feita em sequência (por exemplo, através de WHERE), já que as linhas de uma coluna chave serão armazenadas de acordo com uma ordem;
- Como uma coluna indexada é considerada única em uma tabela, podemos defini-la como **IDENTITY**.

Todos os índices **NonClustered** utilizam os valores chave definidos no índice **Clustered** como chaves de consulta. As entradas do índice **NonClustered** possuem não somente esses valores chave, como também colunas-chave definidas para o próprio índice **NonClustered**.

4.2.3. Índice NonClustered

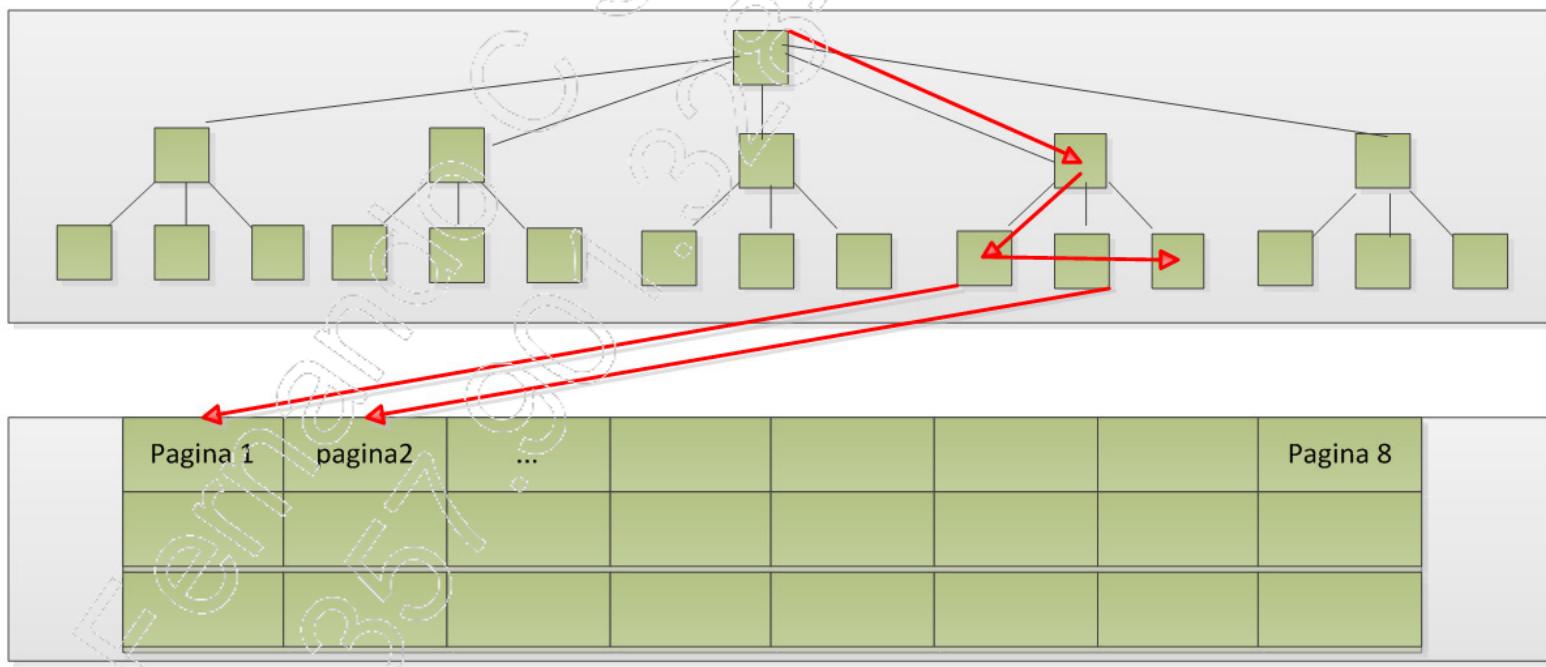
Os dados são ordenados de maneira lógica quando utilizamos o índice **NonClustered**. As chaves de índice e os ponteiros são encontrados no nível folha. Como já foi explicado anteriormente, o índice **Clustered** deve ser criado antes do **NonClustered** para que não seja necessário construí-los novamente.

Este índice é recomendado quando desejamos obter um pequeno conjunto de resultados.

O exemplo a seguir mostra como criar um índice **Nonclustered**:

```
CREATE NONCLUSTERED INDEX I_Funcionario_2  
ON Funcionario(Nome_Func)
```

Representação gráfica para índices não clusterizados:



Através dos índices **NonClustered**, é possível otimizar a utilização das queries não suportadas pelo índice **Clustered**. Cada tabela pode conter até 249 índices **NonClustered**, incluindo as que são criadas com a ajuda das constraints **UNIQUE** e **PRIMARY KEY**. No entanto, os índices XML não são considerados **NonClustered**.

A estrutura de árvore do índice **NonClustered** é praticamente a mesma do **Clustered**, com a diferença de que o nível folha de **NonClustered** possui ponteiros para a página de dados em vez de linhas de dados. Esse nível ainda possui uma coluna indexada.

Caso a tabela não tenha um índice **Clustered**, a classificação das linhas do nível folha de um índice **NonClustered** é feita com base na ordem das colunas que deverão compor esse índice.

As linhas encontradas no nível folha de um índice **NonClustered** possuem um ponteiro para indicação dos dados da coluna que será ordenada, bem como do número da página. No entanto, caso a tabela já possua um índice **Clustered**, cada uma dessas linhas também conterá o valor da coluna-chave do índice **Clustered**, além dos dados da coluna que será ordenada e do número da página. O índice **NonClustered** será lido primeiramente pelo SQL Server para que os dados nas páginas de dados sejam obtidos depois. Essa sequência só não ocorrerá se todas as colunas da cláusula **WHERE** e da lista do **SELECT** fizerem parte desse índice.

Um índice **NonClustered** único será definido quando uma constraint **UNIQUE** for criada. Também é possível definir um índice **Clustered** único em uma tabela que ainda não possui um índice **Clustered**.

O acréscimo de muitas colunas sem necessidade pode comprometer alguns aspectos relacionados à performance. Por exemplo, a quantidade de linhas que poderá fazer parte de uma página será maior, o que exigirá mais espaço em disco para armazenar o índice. Também será necessária uma manutenção periódica e uma avaliação a respeito do andamento da performance do sistema.

4.2.4. Índice Unique

Em um índice Unique (também conhecido como único), a coluna indexada não pode conter valores repetidos. Caso a tabela já contenha dados no momento da criação desse índice, o sistema verificará se alguns desses dados não estão repetidos. A mesma análise será feita a cada nova alteração ou inclusão de uma linha. Se algo duplicado for encontrado, uma mensagem de erro será exibida e o comando será encerrado.

Os valores duplicados podem ser encontrados a partir da execução da seguinte sintaxe:

```
SELECT <idx_col>, COUNT(<idx_col>)
FROM <table_name>
GROUP BY <idx_col>
HAVING COUNT(<idx_col>) > 1
```

Para solucionar o problema referente a valores duplicados, podemos executar um dos procedimentos a seguir:

- Eliminar ou acrescentar colunas de modo que seja possível garantir a integridade dos dados;
- Erros relacionados à entrada de dados devem ser corrigidos manualmente;
- A opção **IGNORE_DUP_KEY** define a especificação do erro responsável pela duplicação de valores de chave em um comando **INSERT** que atua em múltiplas linhas de dados. No caso de **ON**, somente as linhas com valores duplicados serão ignoradas pelo SQL Server Database Engine. No entanto, a configuração **OFF** faz com que todas as demais linhas sejam recusadas. É importante lembrar que **IGNORE_DUP_KEY** não é utilizada em um índice baseado em XML ou em um índice criado em uma view;
- Erros também podem surgir se houver mais de uma linha de dados com valores de chave nulos. Por essa razão, as colunas escolhidas para o índice único ou mesmo para a constraint devem ser definidas como **NOT NULL**.

Como cada linha é definida como única em um índice único, é possível garantir a integridade dos dados existentes na tabela. Uma tabela pode conter diversos índices NonClustered únicos. Vejamos agora as condições de utilização de um índice único quando trabalhamos com constraints:

- Caso a tabela ainda não possua um índice Clustered ou se não foi definido um índice NonClustered único, será criado automaticamente um índice Clustered único assim que a constraint **PRIMARY KEY** for definida. A coluna definida como **PRIMARY KEY** não aceita valores nulos;
- De acordo com o padrão, o índice NonClustered único é criado quando uma constraint **UNIQUE** é definida. No entanto, também podemos definir um índice Clustered único caso a tabela não possua nenhum índice Clustered.

4.2.5. Índice composto

Um índice Clustered, NonClustered ou único que apresenta mais de uma coluna é definido como composto. Trata-se de uma opção recomendada quando devemos pesquisar duas ou mais colunas como uma unidade, principalmente no caso da indexação de chaves estrangeiras compostas ou FK.

Cada uma das colunas que pertencem ao índice composto é ordenada, portanto a ordem é um dos atributos mais importantes quando trabalhamos com esse tipo de coluna. Além disso, todas as colunas de um índice composto devem fazer parte da mesma tabela. É possível reunir até 16 colunas para formar esse tipo de índice.

4.2.6. Índices comprimidos

Índices podem sofrer compressão, assim como as tabelas, possibilitando uma melhor alocação em disco e ocupando menos espaço. A compressão não pode ser aplicada às tabelas de sistema.

Para serem comprimidos, os índices de uma tabela podem ser alterados. A seguir, veremos um exemplo de compressão de índice, mas antes vamos comprimir a tabela:

```
EXEC sp_estimate_data_compression_savings 'dbo', 'aluno', NULL,
NULL, 'ROW' ;

ALTER TABLE dbo.aluno REBUILD PARTITION = ALL
WITH (DATA_COMPRESSION = ROW);
GO
```

Agora, vamos comprimir o índice:

```
SELECT name, index_id
FROM sys.indexes
WHERE OBJECT_NAME (object_id) = N'aluno';

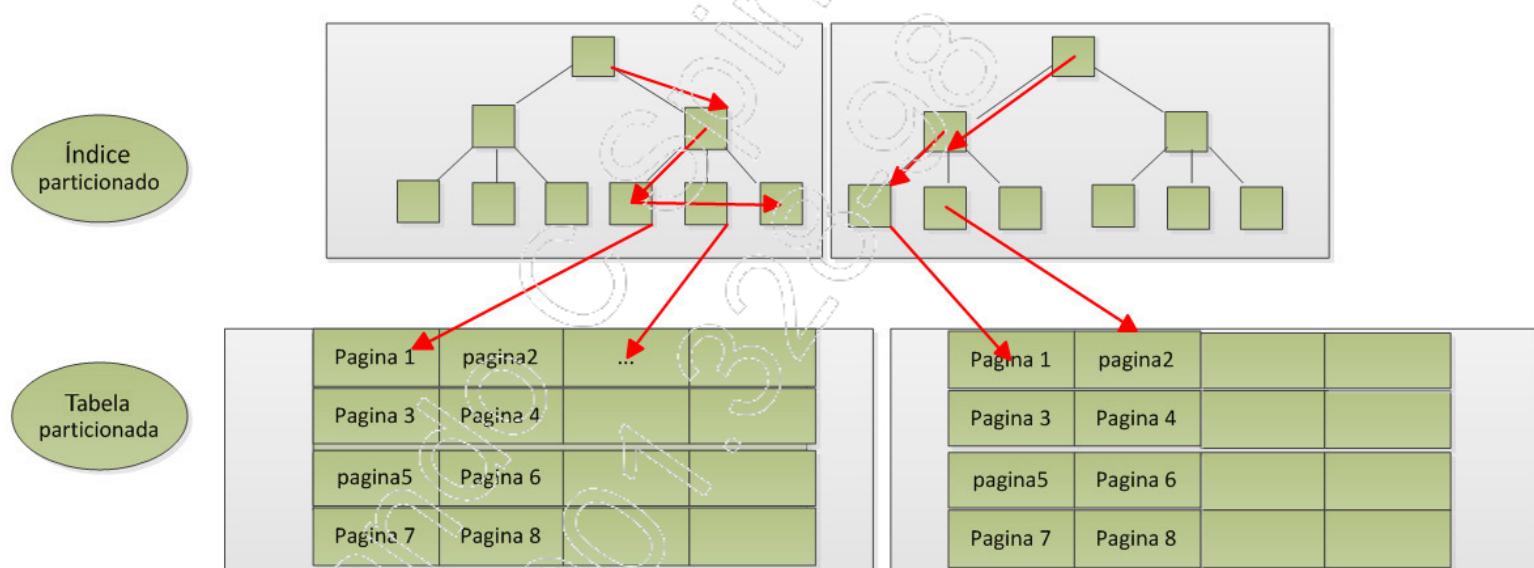
EXEC sp_estimate_data_compression_savings
    @schema_name = 'dbo',
    @object_name = 'aluno',
    @index_id = 1,
    @partition_number = NULL,
    @data_compression = 'PAGE' ;

ALTER INDEX ALUNO_NOME_NI ON DBO.ALUNO REBUILD WITH DATA_COMPRES-
SION = PAGE);
```

4.2.7. Índices particionados

Índices particionados são estruturas de índices (clusterizados ou não clusterizados) que são divididas em partições e cada uma delas faz parte do índice como um todo. No entanto, cada partição pode ter parâmetros e configurações próprias. São recomendados para utilização em tabelas de grande volume, tanto em sistemas transacionais (OLTP) quanto em sistemas de tomada de decisão (DSS).

Geralmente, utiliza-se índice particionado quando há o particionamento da tabela correspondente. A imagem a seguir mostra a estrutura de um índice particionado:



Os benefícios do uso de índices particionados são:

- A leitura de uma partição é mais eficiente do que a leitura de toda a estrutura de um índice. Com isso, diminui o I/O em disco;
- Movimentação das partições de forma independente, facilitando o arquivamento de dados históricos;
- Operação de truncamento da partição da tabela provoca o truncamento da partição correspondente do índice.

4.2.8. Pilhas

Algumas tabelas podem apresentar páginas de dados sem qualquer índice ou que possuem apenas índices do tipo NonClustered. Essas páginas são chamadas de pilhas.

Uma tabela que não possui índice Clustered também é chamada de pilha. De acordo com o padrão, uma pilha possui apenas uma partição, o que não impede de contar com múltiplas partições. Neste caso, cada uma das partições terá uma estrutura adequada para conter os dados necessários. O valor de **index_id** é zero para cada partição, sendo que a pilha possui somente uma linha nas partições sys.

4.3. Determinando a criação de um índice

Para verificar a necessidade de se criar ou não um índice, é importante considerar três aspectos distintos, os quais serão detalhados a seguir:

- **Seletividade**

O número de registros que são retornados por uma query pode ser calculado em relação à quantidade total de registros disponíveis em uma tabela. O percentual obtido pode ser chamado de seletividade.

Caso uma query consiga obter muitos registros como retorno, a seletividade será definida como baixa. Mas se poucos registros forem retornados, esse percentual será alto.

Caso seja necessário otimizar a performance de uma query com seletividade alta, será necessária a criação de um índice. Este índice servirá como referência para a consulta de uma tabela que possui uma grande quantidade de informações, já que muitos registros podem ser retornados.

- **Densidade**

O número de valores repetidos em uma coluna é definido por meio da densidade. Caso exista uma quantidade baixa de valores repetidos na coluna, a densidade será atribuída como baixa. Mas se houver muitos valores repetidos, a densidade será alta.

Caso seja necessário otimizar a performance de uma query com densidade baixa, será necessário criar um índice. Este índice servirá como referência para a consulta de uma tabela que possui uma grande quantidade de informações e poucos valores duplicados na coluna base da pesquisa.

- **Estatísticas**

Tanto a seletividade como a distribuição dos valores-chave dos índices são descritos por meio das estatísticas que fazem parte de todos os tipos de índices. Por meio das estatísticas, podemos avaliar a eficiência de um índice na recuperação de dados que estejam relacionados a um range de valores ou a um valor-chave.

As estatísticas, assim como a seletividade, permite que a navegação através das tabelas e de views indexadas seja otimizada no momento em que os comandos Transact-SQL são processados. Um relatório das estatísticas de um índice pode ser obtido por meio do comando **DBCC_SHOW_STATISTICS**.

As estatísticas são armazenadas da mesma forma que as imagens, ou seja, como uma string longa formada por bits, através de diversas páginas. Os dados são distribuídos conforme a indicação da coluna **statblob** da tabela sysindexes.

As colunas sem índices também podem utilizar as estatísticas, desde que o comando **CREATE STATISTICS** seja executado. O otimizador também pode ser acionado para que as estatísticas sejam criadas.

Por meio do comando **UPDATE STATISTICS**, podemos atualizar as estatísticas com frequência. A atualização das estatísticas também pode ser feita automaticamente pelo SQL Server, que é capaz de averiguar o momento apropriado para realizar essa ação.

Para que as estatísticas sejam criadas, o SQL Server providencia uma análise de alguns dados de uma tabela ou da tabela inteira. Dessa forma, um Plano de Execução Otimizado para uma query poderá ser esquematizado posteriormente.

Em uma estatística, devem constar informações como a quantidade de registros da tabela, a hora e a data da última atualização, a densidade das colunas, o tamanho médio da chave utilizada como base da estatística, a quantidade de etapas necessária para a distribuição e o número de registro na amostra que será utilizada para análise.

As estatísticas podem ser criadas automaticamente pelo SQL Server caso a opção **Auto Create Statistics** do banco de dados esteja configurada como verdadeira. No entanto, elas serão criadas somente para colunas indexadas que possuem dados e as não indexadas que são definidas em cláusulas WHERE e operações JOIN.

É possível, ainda, definir estatísticas de modo manual, desde que elas sejam utilizadas para todas as colunas que fazem parte de um índice composto, com exceção da primeira. Também podemos utilizar essas estatísticas criadas manualmente para colunas não indexadas que servem como referência de pesquisas em cláusulas WHERE.

4.3.1. Criando índices

A partir de agora, apresentaremos diferentes maneiras para criar índices no SQL Server.

4.3.2.Criando índices com comando

Consideremos a seguinte sintaxe:

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX index_name
    ON <object> ( column [ ASC | DESC ] [ ,...n ] )
    [ INCLUDE ( column_name [ ,...n ] ) ]
    [ WHERE <filter_predicate> ]
    [ WITH ( <relational_index_option> [ ,...n ] ) ]
    [ ON { partition_scheme_name ( column_name )
        | filegroup_name
        | default
    }
]
[ FILESTREAM_ON { filestream_filegroup_name | partition_scheme_name | "NULL" } ]

[ ; ]

<object> ::=

{
    [ database_name. [ schema_name ]. | schema_name. ]
    table_or_view_name
}

<relational_index_option> ::=
{
    PAD_INDEX = { ON | OFF }
    | FILLFACTOR = fillfactor
    | SORT_IN_TEMPDB = { ON | OFF }
    | IGNORE_DUP_KEY = { ON | OFF }
    | STATISTICS_NORECOMPUTE = { ON | OFF }
    | DROP_EXISTING = { ON | OFF }
    | ONLINE = { ON | OFF }
    | ALLOW_ROW_LOCKS = { ON | OFF }
    | ALLOW_PAGE_LOCKS = { ON | OFF }
    | MAXDOP = max_degree_of_parallelism
    | DATA_COMPRESSION = { NONE | ROW | PAGE }
        [ ON PARTITIONS ( { <partition_number_expression> | <range>
    }
    [ , ...n ] ) ]
}
```

SQL 2014 - Módulo III

```
<filter_predicate> ::=  
    <conjunct> [ AND <conjunct> ]  
  
<conjunct> ::=  
    <disjunct> | <comparison>  
  
<disjunct> ::=  
    column_name IN (constant ,...n)  
  
<comparison> ::=  
    column_name <comparison_op> constant  
  
<comparison_op> ::=  
    { IS | IS NOT | = | <> | != | > | >= | !> | < | <= | !< }  
  
<range> ::=  
<partition_number_expression> TO <partition_number_expression>
```

Sintaxe mantida por motivos de compatibilidade com versões anteriores do SQL Server 2014

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX index_name  
    ON <object> ( column_name [ ASC | DESC ] [ ,...n ] )  
    [ WITH <backward_compatible_index_option> [ ,...n ] ]  
    [ ON { filegroup_name | "default" } ]  
  
<object> ::=  
{  
    [ database_name. [ owner_name ] . | owner_name. ]  
    table_or_view_name  
}  
  
<backward_compatible_index_option> ::=  
{  
    PAD_INDEX  
    | FILLFACTOR = fillfactor  
    | SORT_IN_TEMPDB  
    | IGNORE_DUP_KEY  
    | STATISTICS_NORECOMPUTE  
    | DROP_EXISTING  
}
```

- **UNIQUE**

Por meio do comando **UNIQUE**, é possível criar um índice único. Enquanto o índice é criado, o SQL Server buscará quaisquer valores repetidos que possam existir na tabela. Caso algum valor desse tipo seja encontrado, o comando não será mais executado.

- **CLUSTERED**

Caso ainda não exista um índice Clustered na tabela, será criado um novo índice, sendo que cada tabela pode apresentar apenas um índice desse tipo.

- **NONCLUSTERED**

Até 249 índices NonClustered podem fazer parte de uma tabela. Por meio deste comando, podemos criar um índice do tipo NonClustered.

- **FILLFACTOR e PAD_INDEX**

O **FillFactor** consiste em um fator de preenchimento relacionado aos índices e páginas de dados. Na estrutura de índice, este fator atinge o nível folha. Também é possível aplicá-lo no nível não-folha por meio da opção **PAD_INDEX**.

Esse fator ainda é capaz de reduzir a ocorrência de splits de página. Além disso, ele consegue disponibilizar um espaço nos índices para permitir expansões eventuais. Este espaço é reservado em uma página de dados, em um índice Clustered ou em uma página de índice, caso seu tipo seja NonClustered.

- **IGNORE_DUP_KEY**

O SQL Server removerá as linhas repetidas que são inseridas em uma tabela que possui índice Clustered único. No entanto, caso esse índice tenha sido criado com a opção **IGNORE_DUP_KEY**, haverá uma falha somente nas linhas que não atendem à exigência de unicidade. O comando **UPDATE** não fará nenhum efeito em uma tabela cujo índice único foi criado com essa opção.

ATENÇÃO! Em colunas que já possuem valores repetidos, não será possível a criação de índices únicos, mesmo que eles sejam definidos através da opção **IGNORE_DUP_KEY**.

Vejamos o exemplo de uma tabela na qual nenhum índice foi colocado:

```
CREATE TABLE Teste_Um
(
    Cod_Testa int NOT NULL,
    Nome_Testa char(20) NOT NULL
)
```

Para inserir dados nessa tabela, temos o seguinte código:

```
INSERT Teste_Um VALUES (1,'Calor')
INSERT Teste_Um VALUES (2,'Vapor')
INSERT Teste_Um VALUES (3,'Temperatura')
INSERT Teste_Um VALUES (3,'Pressão')
```

Agora, criaremos outra tabela que não possui índices:

```
CREATE TABLE Teste_Dois
(
    Cod_Testa int NOT NULL,
    Nome_Testa char(20) NOT NULL
)
```

Utilizaremos a opção **WITH IGNORE_DUP_KEY** para a inserção de um índice único em uma tabela que já possui dados:

```
CREATE UNIQUE CLUSTERED  
INDEX I_TesteUm ON Teste_Um(Cod_Testo)  
WITH IGNORE_DUP_KEY
```

A seguir, temos a resposta obtida a partir da execução realizada anteriormente:

```
/*Server: Msg 1505,Level 16,State 1,Line 1  
CREATE UNIQUE INDEX terminated because a duplicate key  
was found. Most significant primary key is '3'.
```

Suponhamos que um índice será inserido na tabela que não possui dados duplicados:

```
CREATE UNIQUE CLUSTERED  
INDEX I_TesteDois ON Teste_Dois(Cod_Testo)  
WITH IGNORE_DUP_KEY
```

A seguir, temos a resposta obtida a partir da execução realizada anteriormente:

```
**The command(s) completed successfully.*/
```

Em uma tabela cujo índice único foi criado por meio de **WITH IGNOREDUP_KEY**, serão inseridos dados duplicados e também não duplicados:

```
INSERT Teste_Dois  
SELECT*FROM Teste_Um
```

SQL 2014 - Módulo III

A partir da execução anterior, podemos obter o seguinte resultado:

```
**
Duplicate key was ignored.
*/
SELECT * FROM Teste_Dois
/*
Cod_Test Nom_Teste
_____
1 Calor
2 Vapor
3 Temperatura
*/
(3 row(s) affected)
```

Como o índice foi criado com a opção **WITH IGNORE_DUP_KEY**, o SQL Server eliminou somente a linha responsável pela duplicidade, como pudemos observar anteriormente. Quando a última linha de código for executada, a mensagem 3604 será devolvida (refere-se a **Duplicate key was ignored**).

- **DROP_EXISTING**

Quando alteramos um índice Clustered ou NonClustered que já faça parte de uma tabela, devemos utilizar a opção **DROP_EXISTING** no comando **CREATE INDEX**. Dessa forma, é possível recriar o índice conforme as alterações que devem ser realizadas. É importante lembrar que não devemos modificar o nome do índice definido no comando **CREATE INDEX**.

- **STATISTICS_NORECOMPUTE**

As estatísticas dos índices são recalculadas de maneira automática no SQL Server, porém, é possível desativar esse recurso. Para isso, devemos utilizar a opção **STATISTICS_NORECOMPUTE**. Caso seja necessário ativar mais uma vez a atualização automática, basta não utilizar a opção **NORECOMPUTE** ao executar o comando **UPDATE STATISTICS**.

- **ON FILEGROUP**

Permite que o índice seja criado no filegroup especificado.

4.3.3. Criando índices graficamente

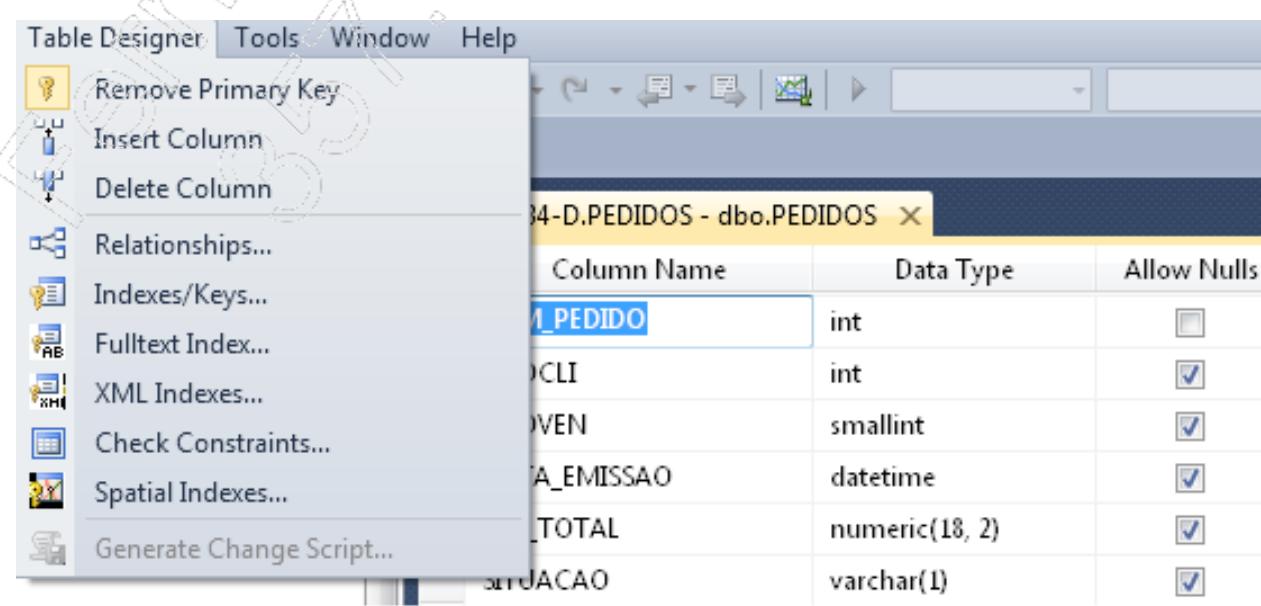
Para criarmos um índice graficamente, devemos seguir os passos adiante:

1. Acesse o **Object Explorer**;
2. Clique com o botão direito do mouse sobre a tabela na qual o índice Clustered será criado;
3. Escolha a opção **Design**;

Será possível observar a tabela na opção **Table Designer**:

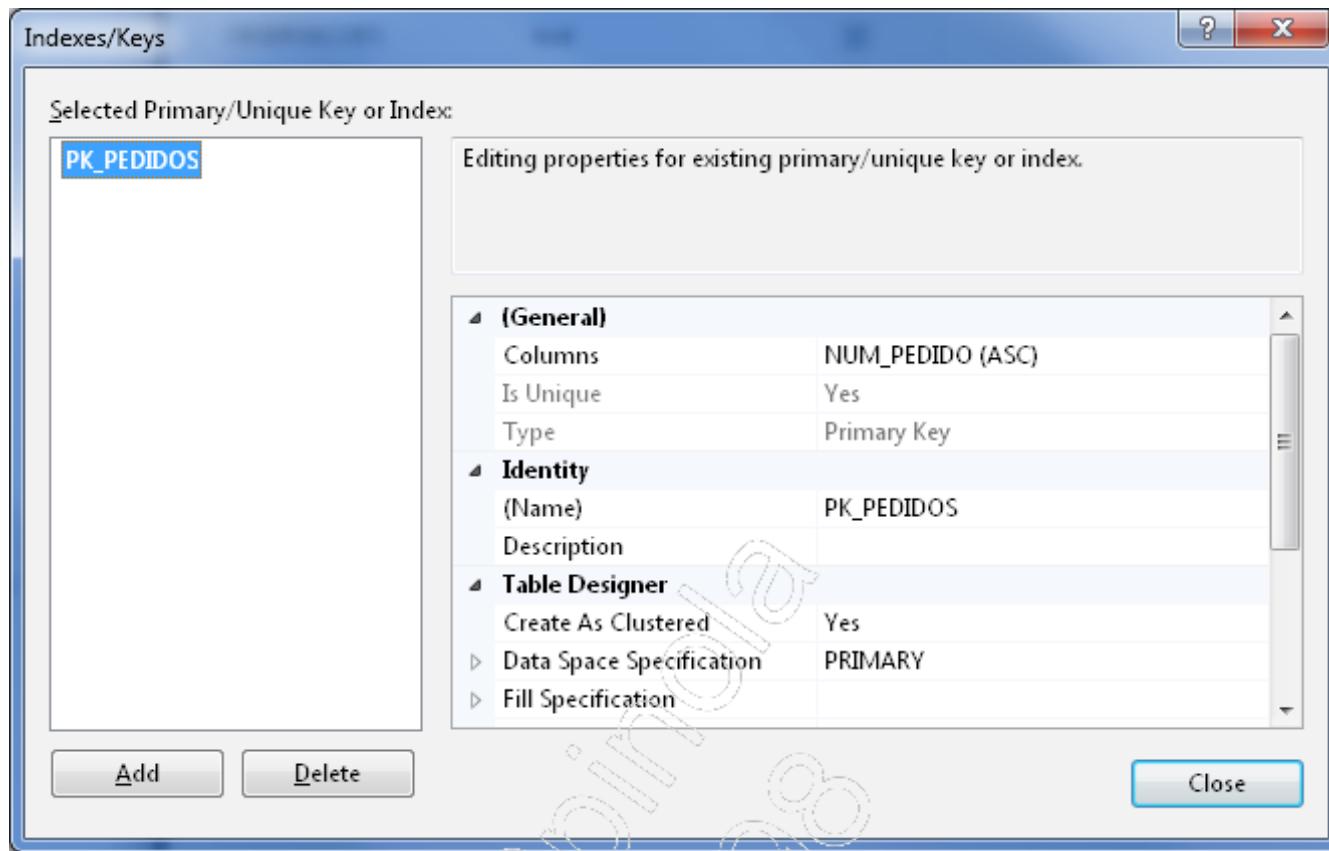
Column Name	Data Type	Allow Nulls
NUM_PEDIDO	int	<input type="checkbox"/>
CODCLI	int	<input checked="" type="checkbox"/>
CODVEN	smallint	<input checked="" type="checkbox"/>
DATA_EMISSAO	datetime	<input checked="" type="checkbox"/>
VLR_TOTAL	numeric(18, 2)	<input checked="" type="checkbox"/>
SITUACAO	varchar(1)	<input checked="" type="checkbox"/>
OBSERVACOES	text	<input checked="" type="checkbox"/>
DATA_ALTERACAO	datetime	<input type="checkbox"/>

4. Em seguida, na barra de menus do SQL Server Management Studio, selecione o menu **Table Designer**:



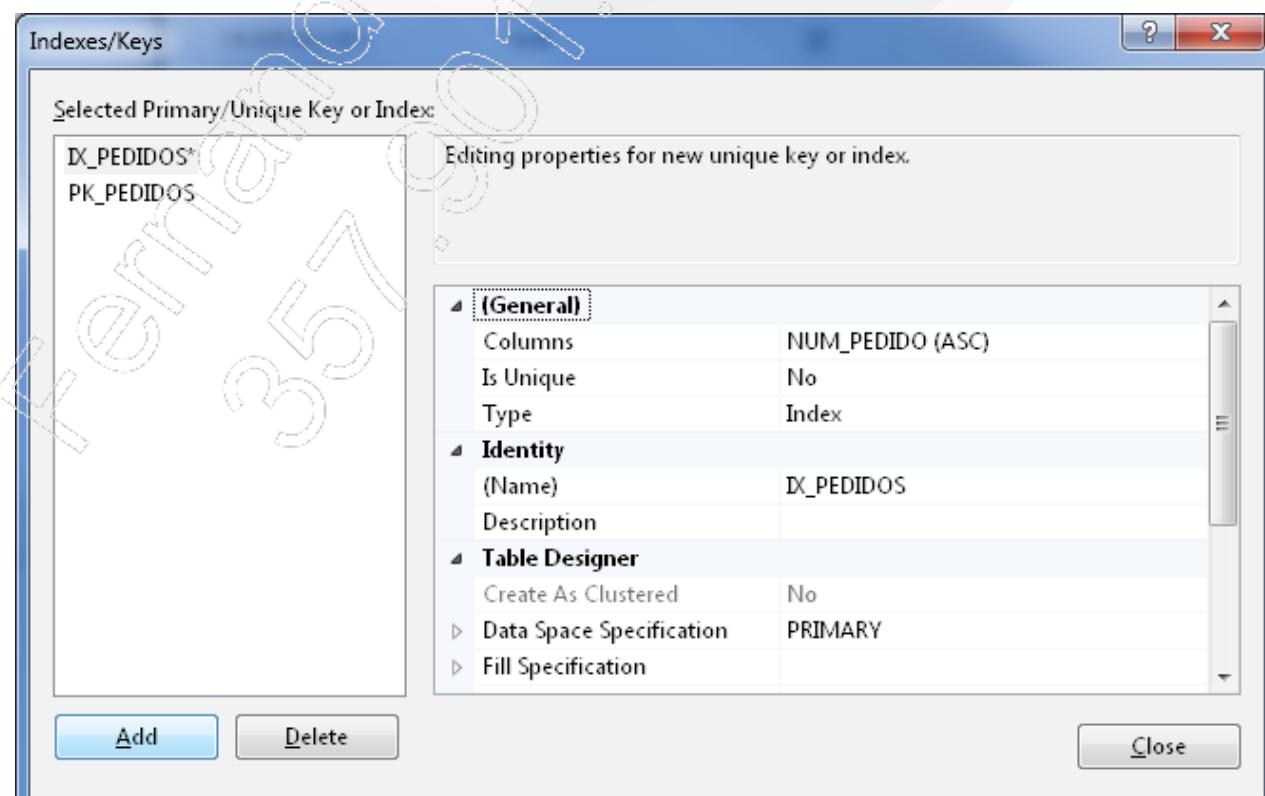
SQL 2014 - Módulo III

5. Selecione a opção **Indexes/Keys** do menu **Table Designer**;



6. Selecione **Add** a partir da caixa de diálogo **Indexes/Keys**;

7. A partir da lista **Selected Primary/Unique Key or Index**, escolha o novo índice;



8. Agora, escolha **Create as Clustered** na grade localizada à direita da lista;
9. À direita da opção selecionada, clique em **Yes** a partir da lista drop-down;
10. Agora basta salvar a tabela para que o índice seja finalmente criado no banco de dados.

4.4. Manutenção de índices

Para realizar a manutenção de um índice que faz parte de uma determinada tabela, devemos seguir diferentes procedimentos, os quais serão detalhados nos subtópicos adiante.

4.4.1. Obtendo informações sobre os índices

Para que seja possível consultar informações sobre os índices que fazem parte de uma determinada tabela, devemos executar a system stored procedure, como mostra o exemplo a seguir:

```
Exec SP_HELPINDEX Funcionario  
sp_helpindex [ @objname = ] 'name'
```

Quando selecionamos esse comando e pressionamos F5 para executá-lo, é possível observar os seguintes conjuntos de resultados:

- **Index_name**: Trata-se do nome de índice, cujo tipo de dado será sysname;
- **Index_description**: Exibe a descrição do índice e o filegroup no qual ele se encontra. Trata-se de um tipo de dado **varchar (210)**. Caso a opção **NORECOMPUTE** de **UPDATE STATISTICS** tenha sido utilizada, essa informação também será apresentada;
- **Index_keys**: Define as views ou colunas utilizadas como base para a construção do índice, sendo que o tipo de dado, neste caso, é **nvarchar(2078)**.

As colunas indexadas poderão aparecer de duas maneiras diferentes: uma refere-se ao padrão, que é a ordem ascendente organizada pelo nome; a outra seria a ordem descendente. Neste caso, o nome da coluna é seguido pelo sinal de menos.

4.4.2. Obtendo informações sobre estatísticas

A função **STATS_DATE()** deve ser utilizada quando desejamos consultar a data em que as estatísticas foram atualizadas. Esta função pode ser aplicada da seguinte maneira:

```
SELECT 'Nome do Índice' = sysindexes.name,  
       'Data da Estatística' = STATS_DATE(sysindexes.  
       id,sysindexes.indid)  
  FROM sysobjects,sysindexes  
 WHERE sysobjects.id = sysindexes.id AND  
   sysobjects.name = 'Funcionario'
```

Os comandos a seguir devem ser executados para a obtenção de informações a respeito das estatísticas dos índices:

- **DBCC SHOW_STATISTICS**

A seguir, temos um exemplo da aplicação deste comando:

```
DBCC SHOW_STATISTICS (Funcionario,PK_Func)  
DBCC SHOW_STATISTICS (Funcionario,F_Funcionario_1)
```

Uma vez executado esse comando, as seguintes colunas serão retornadas se **STAT_HEADER** for especificado:

Nome da coluna	Definição
Nome	Nome da estatística realizada.
Updated	Refere-se à hora e data da última atualização.
Rows	Quantidade de linhas encontradas na tabela.
Rows Sampled	Quantidade de linhas necessárias para as informações estatísticas.
Steps	Quantidade de passos necessários para a distribuição.
Density	Indica a seletividade do prefixo da primeira coluna do índice com exceção de EQ_ROWS .
Average Key Length	Exibe o tamanho médio de todas as colunas que foram indexadas.

Nome da coluna	Definição
String Index	Utilizado apenas nas primeiras colunas baseadas nos tipos de dados char , nchar , varchar , nvarchar , nvarchar(max) , ntext ou text . Caso esteja configurado como YES , a estimativa dos tamanhos possíveis de resultados em condições LIKE é suportada apenas pelas estatísticas que possuem índices de sumário de string.
Filter Expression	Indica o subconjunto de linhas de tabela que está incluso no objeto de estatísticas.
Unfiltered Rows	Indica o total de linhas na tabela antes de aplicarmos Filter Expression .

Caso seja especificado **DENSITY_VECTOR**, as colunas retornadas como resultados após a execução da system stored procedure serão as seguintes:

Nome da coluna	Definição
All Density	Indica a seletividade de vários prefixos de coluna indexada, incluindo EQ_ROWS .
Average Length	Define o tamanho médio de um conjunto de prefixos de colunas indexadas.
Columns	Representa os nomes dos prefixos de colunas indexadas que foram exibidas em Average Length e All Density .

Caso a opção **HISTOGRAM** seja especificada, podemos retornar o seguinte conjunto de resultados:

Nome da coluna	Definição
AVG_RANGE_ROWS	Indica a quantidade média de valores repetidos em um passo do histograma, com exceção do maior valor associado, sendo que RANGE_ROWS / DISTINCT_RANGE_ROWS para DISTINCT_RANGE_ROWS > 0 .
DISTINCT_RANGE_ROWS	Quantidade de valores diferentes em um passo do histograma, com exceção do maior valor.

Nome da coluna	Definição
EQ_ROW	Quantidade de linhas da tabela cujo valor é similar ao maior valor que esteja associado a um passo do histograma.
RANGE_HI_KEY	Indica o maior valor que está vinculado a um passo do histograma.
RANGE_ROWS	Quantidade de linhas que fazem parte de um passo do histograma, com exceção do maior valor associado.

- **SET STATISTICS IO ON/OFF**

As estatísticas de cada comando utilizado podem ser vistas a seguir:

Nome da coluna	Definição
Logical reads	Número de páginas que foram lidas do data cache.
Physical reads	Número de páginas que foram lidas do disco.
Read-ahead reads	Quantidade de páginas que foram adicionadas pela query no cache.
Scan count	Indica o número de leituras que foram feitas.
Table	Define o nome da tabela.
Lob logical reads	Indica a quantidade de páginas text , ntext , image , ou de um tipo de valor extenso, como varchar(max) , nvarchar(max) e varbinary(max) , que foram lidas do data cache.
Lob physical reads	Indica a quantidade de páginas text , ntext , image ou de um tipo de valor extenso que foram lidas do disco.
Lob read_ahead reads	Indica a quantidade de páginas text , ntext , image ou de um tipo de valor extenso que foram alocadas no cache para a query.

- **SET STATISTICS PROFILE**

As informações retornadas são referentes ao perfil de um comando. O número de execuções realizadas em um operador é definido por meio de **Executes**, enquanto **Rows** determina a quantidade total de linhas criadas por meio de cada operador.

Assim que executarmos o comando **SET STATISTICS PROFILE ON** e o comando **SHOWPLAN_ALL**, será possível obter os seguintes retornos:

- **Argument**: Nesta coluna, podemos conferir informações a respeito de uma operação que está sendo realizada. O operador físico utilizado interfere no conteúdo dessa coluna;
- **AvgRowSize**: Define, em bytes, o tamanho médio da linha que será transmitida através do operador;
- **DefinedValues**: Válida apenas para linhas do tipo **PLAN_ROWS**, esta coluna apresenta uma lista de valores separados por vírgulas e calculados a partir de expressões encontradas na query atual ou por meio de valores introduzidos de acordo com uma ordem que permita o processamento da query. É possível referenciar tais valores para outro lugar através da query;
- **EstimateCPU**: Utilizada apenas para linhas do tipo **PLAN_ROWS**, define o custo de CPU possível no caso do operador atual;
- **EstimateExecutions**: Quantidade de vezes em que é executado um operador no momento em que a query atual é rodada;
- **EstimateIO**: Viável apenas para linhas do tipo **PLAN_ROWS**, trata-se do custo de I/O previsto para o operador atual;
- **EstimateRows**: Utilizada somente para linhas do tipo **PLAN_ROWS**, esta coluna indica a quantidade de linhas de saída prevista pelo operador;
- **LogicalOp**: Utilizado apenas para linhas do tipo **PLAN_ROWS**, trata-se de um operador algébrico relacional representado por um nó;

- **NodeId:** Refere-se ao ID do nó na query atual;
- **OutPutList:** As colunas projetadas pela operação atual são separadas por uma vírgula dentro de uma lista;
- **Parallel:** Define se o operador está rodando em paralelo (valor 1) ou não (valor 0);
- **Parent:** Indica o ID do nó do passo pai;
- **PhysicalOp:** Utilizado apenas para linhas do tipo **PLAN_ROWS**, trata-se de um algoritmo de implementação física para o nó;
- **StmtText:** Nesta coluna, encontramos o operador físico e, opcionalmente, o operador lógico. Se as linhas são do tipo **PLAN_ROW**, a descrição da operação poderá ser vista nesta coluna. Caso contrário, será apresentado o texto do comando Transact-SQL. A descrição definida pelo operador físico também pode seguir essa coluna;
- **StmtId:** Indica o número do comando no batch atual;
- **TotalSubtreeCost:** Além de definir o custo possível de um operador, também atribui um custo para os operadores filho;
- **Type:** O tipo de nó será referente ao comando Transact-SQL caso sejam utilizados os nós parentes de cada query. O tipo será **PLAN_ROW** se utilizarmos subnós que representam o plano de execução;
- **Warnings:** Mensagens de alerta relacionadas à mensagem atual são separadas por vírgulas em uma lista. Caso exista uma string **NO STATS: ()**, as estatísticas serão utilizadas como base para que o otimizador de queries possam tomar uma decisão. Caso não existam estatísticas disponíveis, o otimizador de query será obrigado a fazer uma suposição, o que pode levar à escolha de um plano de questão ineficiente. Uma string **MISSING JOIN PREDICATE** indica a utilização de uma **JOIN** sem um predicado **JOIN**, o que afeta diretamente o processamento da query. Em alguns casos, o predicado **JOIN** foi removido acidentalmente, por isso devemos verificar o motivo dessa ocorrência.

- **SET STATISTICS TIME**

Trata-se do tempo utilizado para que a sintaxe seja verificada, bem como para que cada comando seja compilado e executado. Esse valor é expressado em milissegundos.

- **DBCC SHOWCONTIG**

Diversos fatores podem provocar a fragmentação dos dados. Por exemplo, quando adicionamos FillFactor em um índice ou no momento em que dados são excluídos ou mesmo alterados de modo que ocorra sua eliminação.

O sistema OLAP não permite a ocorrência de muitas fragmentações. Já para o sistema OLTP, isso não representa um problema, pois, neste caso, os dados são frequentemente inclusos nas tabelas.

A seguir, temos um exemplo que mostra a utilização do comando **DBCC SHOWCONTIG**:

```
DBCC SHOWCONTIG ('Cliente')
DBCC SHOWCONTIG scanning 'Cliente' table...
Table:'Cliente' (565577053);index ID:1, database ID:8
TABLE level scan performed.
- Pages Scanned :1
- Extents Scanned :1
- Extent Switches :0
- Avg. Pages per Extent :1.0
- Scan Density [Best Count:Actual Count] :100.00% [1:1]
- Logical Scan Fragmentation :0,00%
- Extent Scan Fragmentation :0.00%
Avg. Bytes Free per Page :5002.0
Avg. Page Density (full) :38.20%
DBCC execution completed. If DBCC printed error
messages, contact your system administrator.
```

A seguir, temos a descrição de cada um dos itens apresentados neste comando:

- **Pages scanned:** Quantidade de páginas existentes na tabela;
- **Extents scanned:** Quantidade de extents disponíveis na tabela;
- **Extent Switches:** Quantidade de vezes nas quais o comando DBCC passou entre duas extents;

- **Avg. Pages per Extent:** Quantidade média de páginas que contam com dados por extent;
- **Scan Density [Best Count:Actual Count]:** Caso todos os dados forem adjacentes, seria possível realizar uma contagem com o número ideal de extents. Porém, o número atual de extents na tabela é aquele que realmente indica contagem atual. A fragmentação na tabela fará com que o valor de scan density seja menor do que 100%, pois 100% indica que não houve essa fragmentação;
- **Logical Scan Fragmentation:** Indica a porcentagem de páginas do nível folha que não estão ordenadas. No caso das pilhas, este número não será considerado;
- **Extent Scan Fragmentation:** Indica a porcentagem de extents que não estão ordenadas de acordo com os índices. As pilhas não consideram esse valor;
- **Avg. Bytes Free per Page:** Refere-se ao número de bytes livres encontrados nas páginas de dados. Caso uma pequena parte dessas páginas esteja preenchida, esse número será alto. Ou seja, valores mais baixos são recomendáveis. O tamanho da linha também pode afetar no valor desse número. Quanto maior a linha, maior será o valor do parâmetro;
- **Avg. Page Density (full):** O tamanho da linha é considerado para definir esse valor, referente à densidade média das páginas. A densidade determina o grau de preenchimento das páginas, sendo que valores mais altos são recomendáveis. Para que a fragmentação de uma tabela seja removida, podemos desfragmentar o próprio índice ou reconstruí-lo:

- Desfragmentação do índice:

```
DBCC INDEXDEFRAG ('Impacta','Teste_Um','I_TestUm')
```

- Reconstrução do índice:

```
CREATE INDEX I_TestUm  
ON Teste_Um(Cod_Test)  
WITH DROP_EXISTING
```

4.5. O otimizador e o plano de execução

Para que a execução das queries atinja uma excelente performance, é preciso definir um plano de execução otimizado. A escolha do plano de execução mais apropriado pode ser feita por meio do otimizador de consultas. O plano de execução é armazenado na memória para que ele possa ser utilizado novamente caso haja um novo processamento da query.

O otimizador de consultas também oferece uma melhoria relacionada à utilização dos recursos de computação, uma vez que esse componente é capaz de avaliar o custo estimado para cada um dos planos de execução oferecidos. Caso existam muitos planos à disposição, o otimizador realiza cálculos complexos baseados em algoritmos em vez de analisar todas as combinações viáveis.

Os valores referentes aos custos para obtenção de dados são obtidos por meio de estatísticas de distribuição. Tais estatísticas definem a seletividade de valores de uma coluna ou de um índice específico. Por exemplo, uma coluna com os números de CPF de cada pessoa é mais seletiva do que outra coluna que possui sobrenomes. As estatísticas são fundamentais para a eficiência do trabalho do otimizador de consultas.

O otimizador de consulta também possui um papel importante na obtenção de dados por meio do comando **SELECT**. Isto acontece porque **SELECT** não segue regras específicas para o retorno de dados (ou seja, trata-se de um comando não-procedural). Por meio de um processo de otimização, o SQL Server analisa primeiramente esse comando para definir qual será o método mais apropriado.

Para realizar a otimização, o otimizador de query baseia-se não apenas na query, como também nas estatísticas de banco de dados e nas definições de índice e de tabela. O resultado será a obtenção de um plano de execução de query.

Um plano de execução é composto de vários passos a serem seguidos para que uma query seja executada. Cada um dos passos é descrito por meio de operadores lógicos e físicos.

O modo como será feita a execução da query é descrito pelo operador lógico, que também informa a operação relacional algébrica necessária para o processamento da query. Já a implementação do algoritmo físico utilizado nesse processamento será descrita por meio do operador físico.

Em um comando **SELECT**, podemos utilizar diferentes cláusulas, como:

- Cláusula **FROM**: Informa a tabela na qual estão armazenados os dados de origem;
- Cláusulas **GROUP BY** e **ORDER BY**: Responsáveis pela definição do formato do conjunto de resultados, juntamente com a lista de seleção;
- Cláusulas **ON** e **WHERE** seguidas pela cláusula **FROM**: Exibe as especificações de associação (join) existentes entre as tabelas;
- Cláusulas **HAVING** e **WHERE**: Especificam a qualificação necessária para as linhas das tabelas de origem.

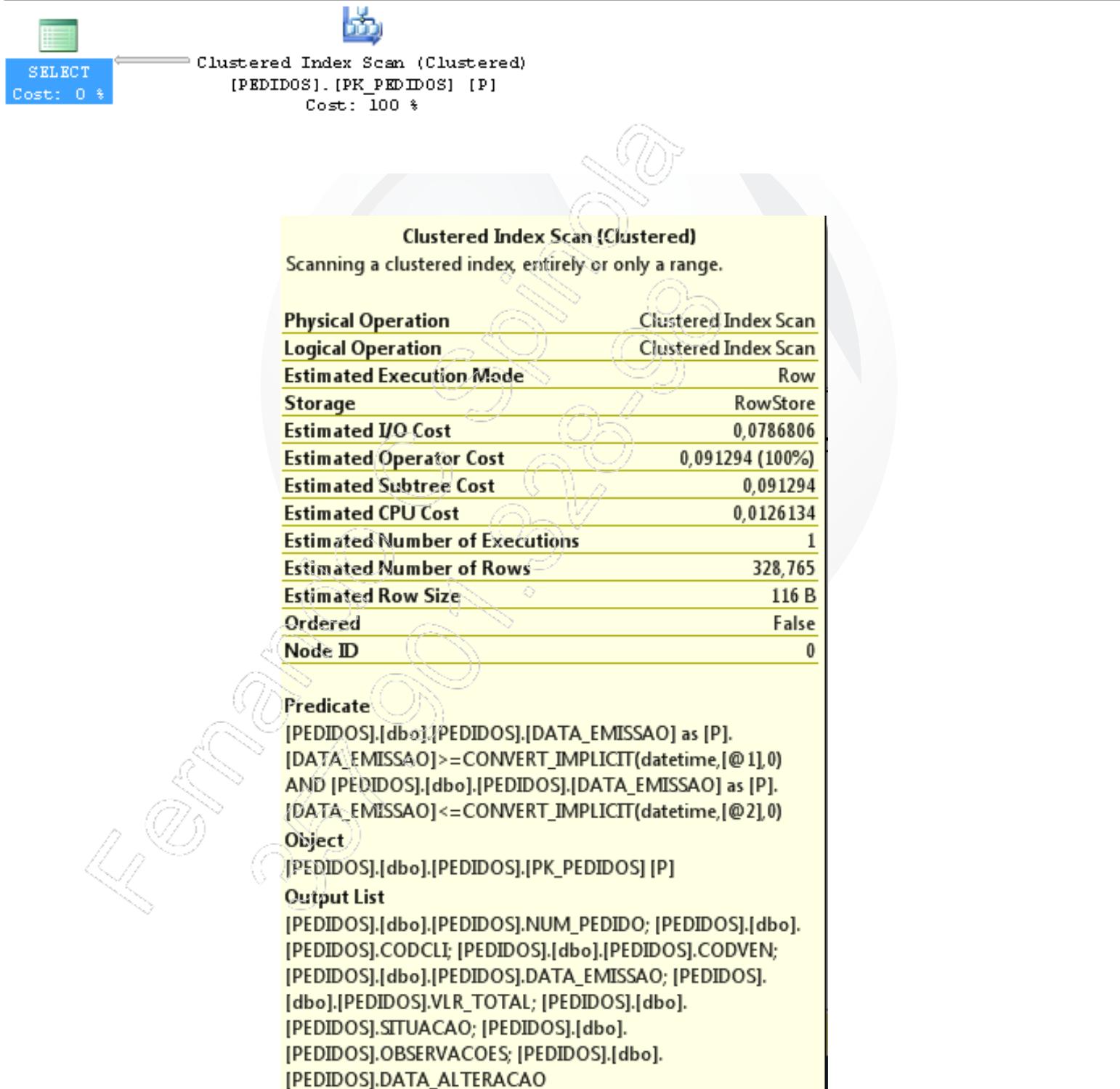
Um plano de execução de consultas possibilita a escolha de diferentes sequências de acesso às tabelas utilizadas como referência para que o servidor possa retornar um conjunto de resultados. Por exemplo, caso sejam especificadas duas tabelas pelo comando **SELECT** (como **Tab1** e **Tab2**), o SQL Server poderá acessar primeiramente a **Tab1** para obter os dados associados na **Tab2**, ou vice-versa.

O método de acesso a dados também é definido pelo plano de execução de query de acordo com os valores de chave existentes. Um índice pode ser utilizado no caso de uma tabela que possui poucas linhas com valores chave específicos. Uma pesquisa de tabela já é mais recomendada quando todas as linhas de uma tabela serão utilizadas como referência ou quando temos uma tabela pequena. Uma pesquisa de índice, por sua vez, deve ser utilizada se as colunas chave de um índice estiverem definidas com a utilização de **ORDER BY**, mesmo se todas as linhas forem requisitadas.

4.5.1. Exemplo de saída de um plano de execução

A seguir, podemos visualizar como um plano de execução é exibido de maneira gráfica. Para que isso seja possível, devemos clicar no ícone **Display Estimated Execution Plan**. Com isso, será exibida a tela a seguir:

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM PEDIDOS AS P WHERE P.DATA_EMISSAO BETWEEN '2007.1.1' AND '2007.1.31'
```



4.5.2. Saídas do plano de execução

Para que as informações de estatísticas retornadas pelo SQL Server possam ser analisadas, devemos utilizar o seguinte comando:

```
SET SHOWPLAN_ALL ON/OFF
```

Quando executamos o comando **SET SHOWPLAN_ALL ON** e, em seguida, qualquer outro comando SQL, podemos obter um retorno que apresenta os seguintes itens:

- **Argument**: Nesta coluna, podemos conferir informações a respeito de uma operação que está sendo realizada. O operador físico utilizado interfere no conteúdo dessa coluna;
- **AvgRowSize**: Define, em bytes, o tamanho médio da linha que será transmitida através do operador;
- **DefinedValues**: Esta coluna apresenta uma lista de valores separados por vírgulas e calculados a partir de expressões encontradas na query atual ou por meio de valores introduzidos de acordo com uma ordem que permita o processamento da query;
- **EstimateCPU**: utilizada apenas para linhas do tipo **PLAN_ROWS**, define o custo de CPU possível no caso do operador atual;
- **EstimateExecutions**: Quantidade de vezes em que é executado um operador no momento em que a query atual é rodada;
- **EstimateIO**: Viável apenas para linhas do tipo **PLAN_ROWS**, trata-se do custo de I/O previsto para o operador atual;
- **EstimateRows**: Utilizada somente para linhas do tipo **PLAN_ROWS**, esta coluna indica a quantidade de linhas de saída prevista pelo operador;
- **LogicalOp**: Utilizado somente para linhas do tipo **PLAN_ROWS**, trata-se de um operador algébrico relacional representado por um nó;

- **NodeId:** Define o ID do nó na query atual;
- **OutPutList:** As colunas projetadas pela operação atual são separadas por uma vírgula dentro de uma lista;
- **Parallel:** Define se o operador está rodando em paralelo (valor 1) ou não (valor 0);
- **Parent:** Define o ID do nó do passo pai;
- **PhysicalOp:** Utilizado apenas para linhas do tipo **PLAN_ROWS**, trata-se de um algoritmo de implementação física para o nó;
- **StmtText:** Nesta coluna, que possui o operador físico e eventualmente o operador lógico, encontramos a descrição da operação ou o texto do comando Transact-SQL que foi executado. Uma descrição definida pelo operador físico também pode surgir logo após essa coluna;
- **StmtId:** Indica o número do comando no batch atual;
- **TotalSubtreeCost:** Além de definir o custo possível de um operador, também atribui um custo para os operadores filho;
- **Type:** O tipo de nó será referente ao comando Transact-SQL caso sejam utilizados os nós parentes de cada query. O tipo será **PLAN_ROW** se utilizarmos subnós que representam o plano de execução;
- **Warnings:** Mensagens de alerta relacionadas à mensagem atual são separadas por vírgulas em uma lista.

Quando executamos o comando **SET SHOWPLAN_TEXT ON** e, em seguida, qualquer outro comando SQL, podemos obter um retorno que apresenta o item **StmtText**, já descrito anteriormente.

4.5.3. Operadores lógicos e físicos

A partir de agora, apresentaremos a descrição dos operadores lógicos e físicos que podem ser utilizados no SQL Server 2014:

- **Sinal de mais (+):** Este operador realiza a concatenação de vários strings binários ou caracteres por meio de uma expressão de string, além de permitir a combinação de nomes de strings e colunas em uma única expressão;
- **Assert:** Trata-se de um operador físico cuja função é verificar uma determinada condição. Uma linha de entrada será validada pelo operador se o retorno de uma expressão for um valor NULL. Caso contrário, um erro será emitido;
- **Bookmark Lookup:** Este operador utiliza o label de um bookmark para buscar a linha correspondente em um índice **Clustered** ou em uma tabela cujo nome encontra-se na coluna **Argument**. O label do bookmark também aparece nesta coluna. Para otimizar essa busca, é recomendável utilizar prefetching assíncrono (read-ahead) caso a cláusula **PREFETCH** surja na coluna **Argument**. A funcionalidade desse operador é garantida pelos operadores **RID Lookup** e **Clustered Index Seek** no SQL Server 2008, já que o **Bookmark Lookup** não é utilizado nesta versão;
- **Clustered Index Delete:** Este operador físico permite que as linhas do índice **Clustered** definidas na coluna **Argument** sejam eliminadas. Para que sejam removidas apenas as linhas que atendem ao predicado **WHERE:()**, devemos incluí-lo na coluna **Argument**;
- **Clustered Index Insert:** Para acrescentar linhas no índice **Clustered** (definido na coluna **Argument**), devemos utilizar esse operador físico. O valor de cada coluna pode ser configurado por meio do predicado **SET:()** dentro da coluna **Argument**. A linha que será inserida ainda pode ser obtida por meio do próprio operador caso este não possua filhos para essa finalidade;
- **Clustered Index Update:** Este operador físico e lógico realiza a atualização dos índices **Clustered** definidos na coluna **Argument**. Valores específicos podem ser determinados por meio do predicado **SET:()**. Podemos referenciar tais valores na query utilizada, dentro do operador ou como uma especificação de **SET**. Para listar os valores definidos pelo operador, devemos utilizar o predicado **DEFINE:()**;

- **Clustered Index Scan:** Por meio deste operador lógico e físico, é realizada uma verificação do índice **Clustered** definido na coluna **Argument**. Caso a cláusula **ORDERED** faça parte da coluna **Argument**, a ordem das linhas retornadas será a mesma do índice **Clustered**. Essa ordenação não será mantida se **ORDERED** não for utilizada, embora a verificação seja realizada de forma otimizada. Também podemos utilizar o predicado **WHERE:()** para que sejam retornadas apenas as linhas que atendam a essa especificação;
- **Clustered Index Seek:** Este operador físico e lógico permite o retorno de linhas do índice **Clustered**. Tanto o nome desse índice como o predicado **SEEK:()** fazem parte da coluna **Argument**. Dessa forma, serão retornadas apenas as linhas que atendem a esse predicado. Embora não seja obrigatório, o predicado **WHERE:()** também pode ser utilizado. A ordenação das linhas definida pelo índice **Clustered** será mantida se a cláusula **ORDERED** fizer parte da coluna **Argument**. Porém, seu uso pode comprometer a eficiência na obtenção dos resultados desejados;
- **Collapse:** Por meio desse operador lógico e físico, podemos dividir a atualização em dois processos diferentes: inserção e remoção de linhas de dados. Esse procedimento é útil quando existem linhas próximas que adicionam e removem os mesmos valores-chave. Também podemos definir uma lista de colunas chave por meio da cláusula **GROUP BY:()** das colunas **Argument**;
- **Compute Scalar:** Este operador físico e lógico permite que um valor escalar computado seja criado por meio da avaliação de um comando. Este valor pode ser referenciado em qualquer parte da query ou retornado para o usuário. O elemento **RunTimeInformation** nem sempre faz parte de **Computer Scalar**, principalmente quando este operador é exibido em **Showplans** gerados a partir de **SET STATISTICS XML**. Neste caso, outros operadores realizarão a função desse operador no plano de query em tempo de execução;
- **Constant Scan:** Tanto uma como várias linhas constantes podem ser inseridas em uma query por meio desse operador. Podemos, ainda, acrescentar colunas nas linhas especificadas com a ajuda do operador **Compute Scalar**;

- **Hash Match:** Com base na entrada de valores, este operador físico calcula o valor hash de cada linha para criar uma tabela hash. Na coluna **Argument**, podemos verificar as colunas utilizadas como referência para a criação dos valores hash dentro de uma lista definida pelo predicado **HASH:()**. Esses valores hash são calculados com base na análise de dada linha de entrada. Em seguida, é feita uma associação de tais valores na tabela hash;
- **Hash Match Root:** Todos os operadores do grupo **Hash Match Team** que encontram-se abaixo do operador **Hash Match Root** são controlados por este. A estratégia de particionamento e uma função hash em comum são compartilhadas pelo operador **Hash Match Root** e por todos os demais do grupo que estão abaixo dele. No SQL Server 2000 SP1 e em outras versões superiores, o otimizador de query já não possui esse operador;
- **Hash Match Team:** Este operador físico integra um grupo de operadores hash que compartilham uma estratégia de particionamento, bem como uma função hash em comum. No SQL Server 2000 SP1 e em outras versões superiores, o otimizador de query já não possui esse operador;
- **Insert:** Como operador lógico, sua finalidade é inserir cada linha a partir de sua entrada no objeto especificado na coluna **Argument**. Os operadores **Table Insert**, **Index Insert**, **Clustered Index Insert** podem ser utilizados como operadores físicos;
- **Merge Join:** Por meio deste operador físico, podemos executar diferentes operações, como union, left semi join, left outer join, left anti semi join, right anti semi join, right outer join, right semi join e inner join. Caso seja realizada uma associação muitos-para-muitos, esse operador contará com o predicado **MANY-TO-MANY MERGE:()** na coluna **Argument**. No entanto, uma associação um-para-muitos utiliza um predicado **MERGE:()**. Nessa coluna, ainda aparecerá uma lista de colunas envolvidas na separação e separadas por vírgulas. Também é importante ressaltar que dois registros devem ser ordenados em suas colunas por meio da inclusão de operações de ordenação no plano de query;

- **Nested Loops:** Este operador físico é responsável pela execução de operações lógicas como left semi join, left outer join, left anti semi join e inner join. Por meio de um índice, é realizada uma busca na tabela interna para cada linha da tabela externa. Também é possível ordenar a tabela externa para que os índices sejam localizados de acordo com a entrada interna. O predicado da coluna **Argument** decidirá se uma determinada linha será considerada aplicável ou não;
- **Parallelism:** Este operador físico permite a execução de operações lógicas relacionadas a streams, como gather streams, distribute streams e repartition streams. Nas colunas **Argument**, poderão constar dois predicados diferentes: **PARTITION COLUMNS:()** com uma lista de colunas que serão particionadas, além de **ORDER BY:()**, responsável pela manutenção da ordem das colunas;
- **Parameter Table Scan:** Utilizado geralmente em queries **INSERT** dentro de uma stored procedure, esse operador físico e lógico realiza uma análise de uma tabela que funciona como um parâmetro na query atual;
- **Remove Scan:** Um objeto remoto, cujo nome aparece na coluna **Argument**, é verificado por meio desse operador físico e lógico;
- **Row Count Spool:** Este operador físico é utilizado para avaliar a existência de linhas e não de dados. Para isso, **Row Count Spool** realiza a contagem de linhas disponíveis para retornar esse número sem a inclusão de dados;
- **Sequence:** As entradas (geralmente atualizações de um objeto diferente) são executadas de cima para baixo, de acordo com uma sequência. Somente as linhas obtidas a partir da entrada mais recente são retornadas quando utilizamos esse operador físico e lógico;
- **Sort:** Todas as linhas introduzidas são ordenadas por meio desse operador lógico e físico. Para que os valores repetidos sejam eliminados, um predicado **DISTINCT ORDER BY: ()** deve constar na coluna **Argument**. Este predicado pode, ainda, exibir uma lista de colunas que serão ordenadas. A ordem descendente é indicada pelo prefixo **DESC**, enquanto a ascendente é representada pelo **ASC**;

- **Stream Aggregate:** Além de agrupar várias colunas, este operador físico realiza o cálculo de uma ou mais expressões agregadas que são referenciadas ou retornadas pela query. Por meio do operador **Stream Aggregate**, as colunas ordenam as entradas de acordo com os grupos definidos. Uma vez calculadas, as expressões agregadas aparecem na coluna **Argument** do planejamento de execução gráfica ou em uma lista na coluna **Defined Values** do resultado do comando **SHOWPLAN_ALL**. No entanto, se o operador for utilizado para agrupar colunas, tanto uma lista dessas colunas, como o predicado **GROUP BY:()**, serão exibidos na coluna **Argument**;
- **Table Insert:** Por meio deste operador físico, é possível acrescentar linhas correspondentes à entrada de valores dentro da tabela definida na coluna **Argument**. O valor de configuração para cada coluna é indicado pelo predicado **SET: ()**, que também faz parte da coluna **Argument**;
- **Table Scan:** Corresponde a um operador físico e lógico capaz de retornar todas as linhas definidas na coluna **Argument**. Também podemos especificar os valores retornados por meio da inclusão do predicado **WHERE: ()** na coluna **Argument**;
- **Table Spool:** Este operador físico faz com que sejam criadas cópias das linhas correspondentes às entradas de dados em uma tabela spool oculta, que é armazenada temporariamente no banco de dados tempdb e que existe apenas durante a consulta;
- **Table Update:** Na tabela definida na coluna **Argument**, as linhas correspondentes à entrada de dados são atualizadas por meio desse operador. O valor de cada uma das colunas atualizadas é determinado pelo predicado **SET:()** e deve ser referenciado na query, em qualquer parte do operador ou pela cláusula **SET**;
- **Top:** Este operador físico e lógico indica o primeiro valor referente à quantidade ou porcentagem de linhas, que pode ser baseada na ordem em que os dados estão classificados. Por meio de **Top**, também podemos garantir limites para a contagem de linhas. As colunas verificadas pelo operador podem fazer parte de uma lista existente na coluna **Argument**. Com relação ao operador **Parallelism**, já apresentado, também é importante considerar que, quando ele não é utilizado no planejamento de query, mas somente na sua compilação, podem ocorrer duas situações diferentes.

Para começar, nem sempre o elemento **RunTime Information** aparece no resultado **Showplan** obtido a partir da opção **Include Actual Execution Plan** do SQL Server ou por meio da execução de **SET STATISTICS XML**.

Isso acontece quando uma query é compilada como paralela, mas é executada como uma query serial durante o tempo de execução. Também há a probabilidade de que zeros sejam exibidos no resultado de **SET STATISTICS PROFILE** em relação à quantidade de execuções e à contagem de linhas.

Caso o servidor esteja sobrecarregado, os planejamentos de query paralelos podem ser executados de modo serial.

4.5.4. Sobrepondo o otimizador

Um índice que será utilizado em uma query pode ser apontado, embora esse procedimento não seja recomendado. Para que o índice seja apontado, devemos digitar o nome do índice que será utilizado pelo SQL na área **Optimizer Hints**. Isso faz com que o otimizador não seja acionado, como podemos ver no comando a seguir:

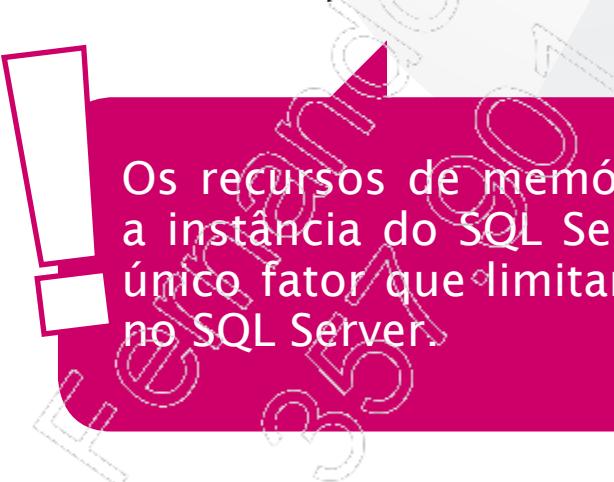
```
SELECT*FROM Cliente  
WITH (INDEX = I_Clientes_3)
```

4.6. Índices Full-Text

O serviço **Microsoft Full-Text Engine for SQL Server**, ou MSFTESQL, mantém um tipo especial de índice funcional baseado em token denominado índice Full-Text, criado pelo próprio serviço.

Na construção de um índice Full-Text, o serviço MSFTESQL cria uma estrutura de índice comprimida, invertida e em pilhas baseada em tokens individuais do **texto** que está sendo indexado. Já na criação de outros tipos de índice, normalmente são construídas estruturas de árvore em B que se baseiam em um valor armazenado em uma linha particular.

O procedimento para criar e manter um índice Full-Text é denominado população de índice. Existem três tipos de população de índice Full-Text suportados pela Microsoft: **Full Population** (População completa), **Change tracking-based population** (População baseada no rastreamento de alterações) e **Incremental timestamp-based population** (População incremental baseada em timestamp). Cada um desses tipos será descrito a seguir.



Os recursos de memória disponíveis da máquina na qual a instância do SQL Server é executada dizem respeito ao único fator que limitará o tamanho de um índice full-text no SQL Server.

4.6.1. Full Population

No momento de uma população completa de um catálogo Full-Text, as entradas de índice são criadas para todas as linhas, em todas as tabelas que o catálogo cobre.

A população completa normalmente ocorre na primeira vez que um índice Full-Text ou catálogo Full-Text é populado. Feito isso, poderemos utilizar as populações incrementais ou populações de rastreamento de alterações para manter os índices.

Contudo, podemos fazer com que um índice Full-Text não seja populado no momento de sua criação. Para isso, podemos realizar as seguintes ações:

1. Utilize o comando **CREATE FULLTEXT INDEX**;
2. Junto da opção **CHANGE TRACKING OFF**, especifique **NO POPULATION**.

Para popular o índice Full-Text, será preciso executar o comando **ALTER FULLTEXT INDEX** com uma das seguintes cláusulas:

- **INCREMENTAL**;
- **UPDATE POPULATION**;
- **START FULL**.

4.6.2. Change Tracking Based Population

Ao especificar a opção **WITH CHANGE_TRACKING**, no comando **CREATE FULLTEXT INDEX**, é possível iniciar o rastreamento das alterações ocorridas em linhas modificadas de uma tabela configurada para indexação Full-Text, linhas das quais o SQL Server mantém um registro. É importante ressaltar que as alterações são refletidas no índice Full-Text.

A maneira como as alterações são propagadas para o índice Full-Text pode ser definida quando utilizamos o rastreamento de alterações.

As alterações podem ser propagadas para o índice Full-Text de maneira manual ou automática. Para propagá-las manualmente, utilizamos a opção **MANUAL**, por meio da qual podemos propagar as alterações conforme um agendamento, ou por meio do **SQL Server Agent**, ou, ainda, por conta própria. Já para a propagação automática, contamos com a opção **AUTO**, a ser utilizada em um dos seguintes comandos:

- **CREATE FULLTEXT INDEX;**
- **ALTER FULLTEXT INDEX.**

A utilização da população baseada no rastreamento de alterações envolve, também, os seguintes aspectos:

- É necessária uma população inicial do Full-Text em utilização;
- Para que o SQL Server não realize o rastreamento de alterações, o usuário deve utilizar a opção **CHANGE TRACKING OFF**. É importante ressaltar que o rastreamento de alterações está relacionado com a ocorrência de um pequeno overhead.

4.6.3. Incremental Timestamp-Based Population

É importante atentar para o fato de que a inexistência de uma coluna de tipo de dado **timestamp** na tabela indexada impossibilitará a utilização da população incremental. Teremos uma população completa caso façamos uma solicitação de população incremental em uma tabela sem uma coluna desse tipo.

Para que um índice full-text seja atualizado com linhas removidas, acrescentadas ou alteradas durante ou após a última população, utilizamos a população incremental.

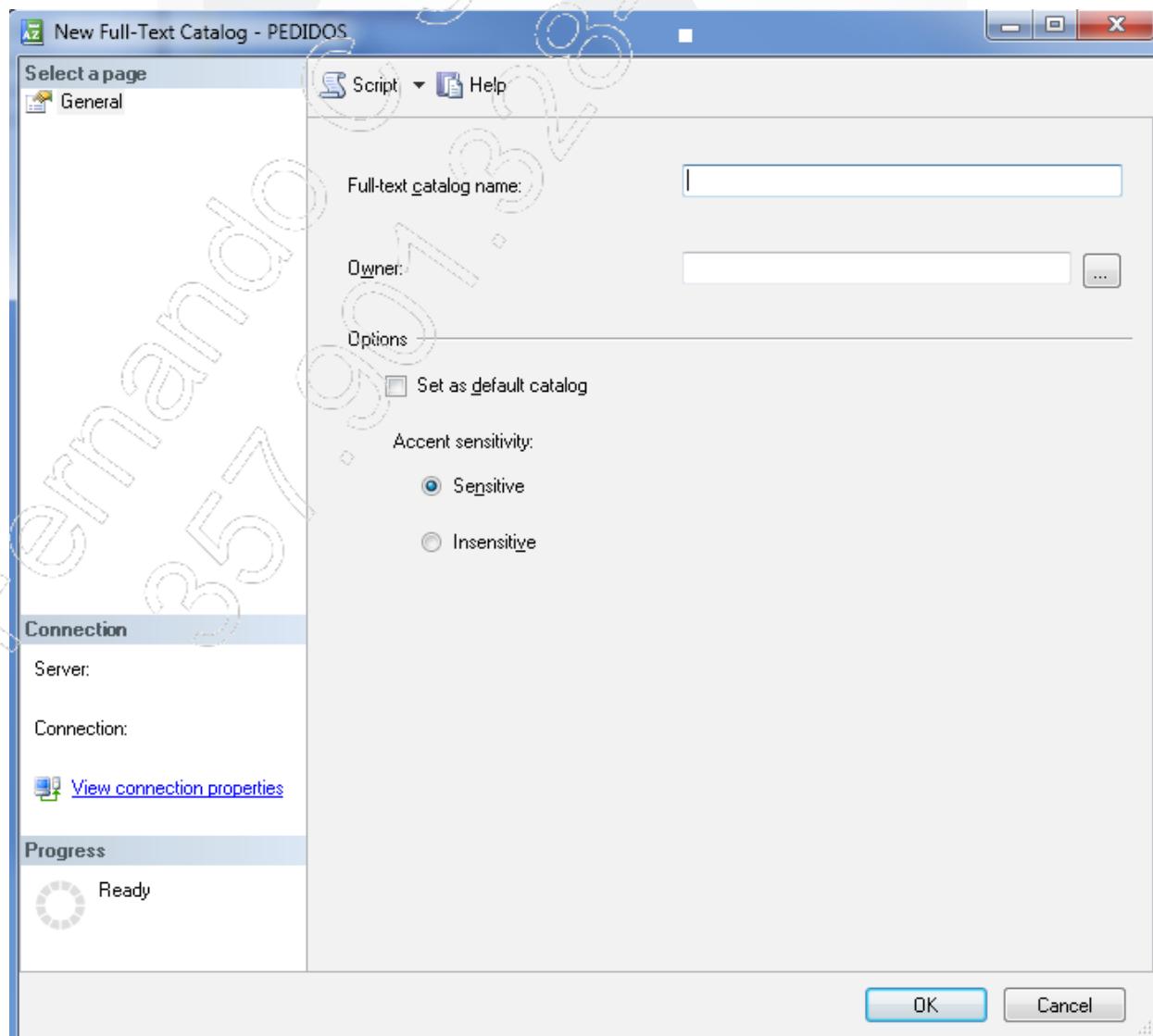
Se algum metadado que afeta o índice Full-Text da tabela foi alterado desde a última população, as solicitações de população incremental são implementadas como populações completas. Essas alterações incluem a modificação de índices, colunas ou definições de índice Full-Text.

Assim que uma nova população incremental for iniciada, será utilizado um valor **timestamp** registrado pelo **SQL Gatherer** ao final de uma população e que é idêntico ao maior valor **timestamp** visto pelo **SQL Gatherer**.

4.6.4. Criando um catálogo FULL-TEXT

Os procedimentos a seguir explicam como criar um catálogo para os índices Full-Text:

1. Abra o **Enterprise Manager**;
2. Expanda o grupo de servidores desejado;
3. Expanda o servidor desejado;
4. Expanda a pasta **Databases**;
5. Expanda o banco de dados desejado;
6. Expanda a pasta **Storage**;
7. Clique com o botão direito do mouse sobre **Full_Text Catalogs**;
8. Selecione **New Full-Text Catalogs**. A seguinte caixa de diálogo será aberta:



4.7. Criando um índice FULL-TEXT

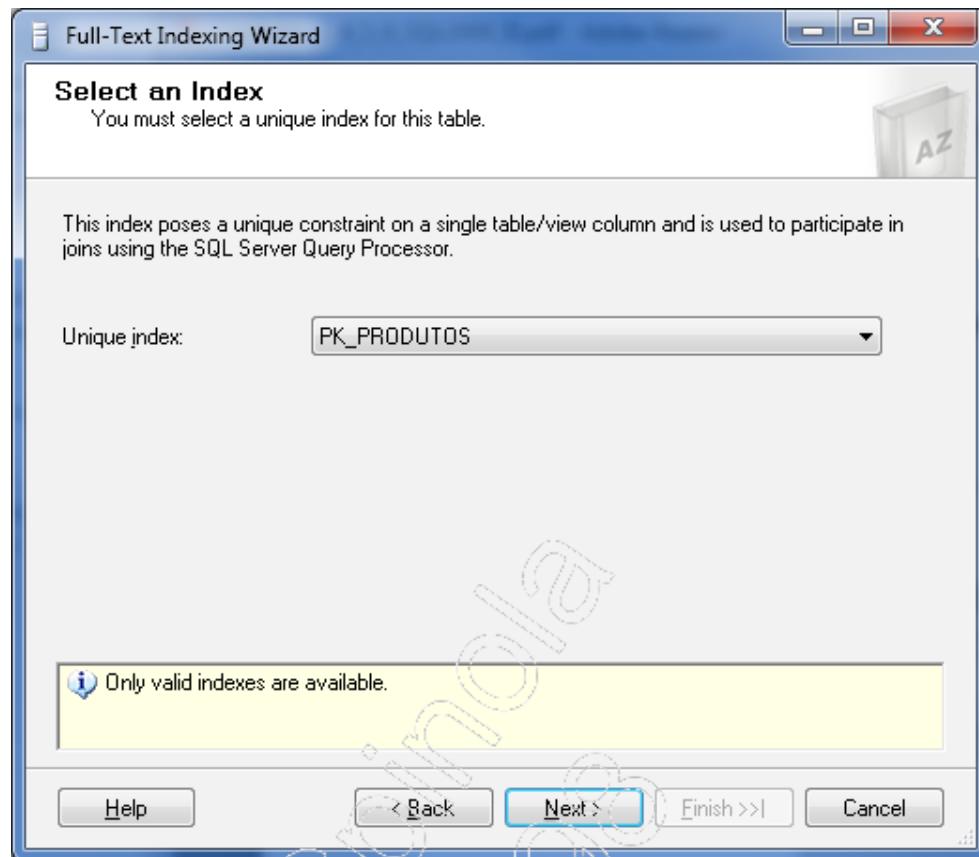
Os procedimentos a seguir explicam como criar um índice full-text:

1. Expanda o grupo de servidores desejado, no SQL Server;
2. Expanda o servidor desejado;
3. Expanda a pasta **Databases**;
4. Expanda ao banco de dados desejado;
5. Expanda a pasta **Tables**;
6. Clique com o botão direito do mouse sobre a tabela desejada;
7. Selecione **Full-Text Index** e, em seguida, **Define Full-Text Index**. A seguinte tela será exibida:

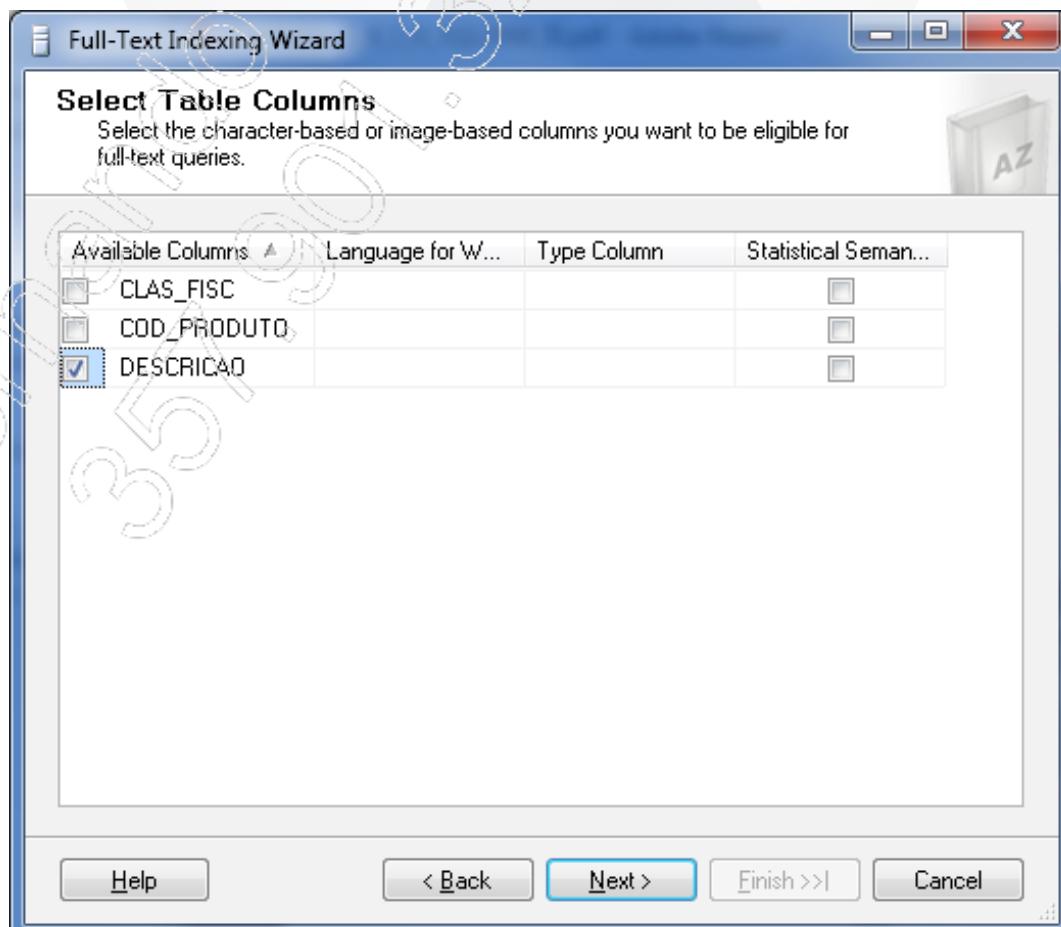


SQL 2014 - Módulo III

8. Clique em **Next**:



9. Na tela seguinte, selecione o índice único já existente na tabela e que possibilitará a criação do índice full-text:

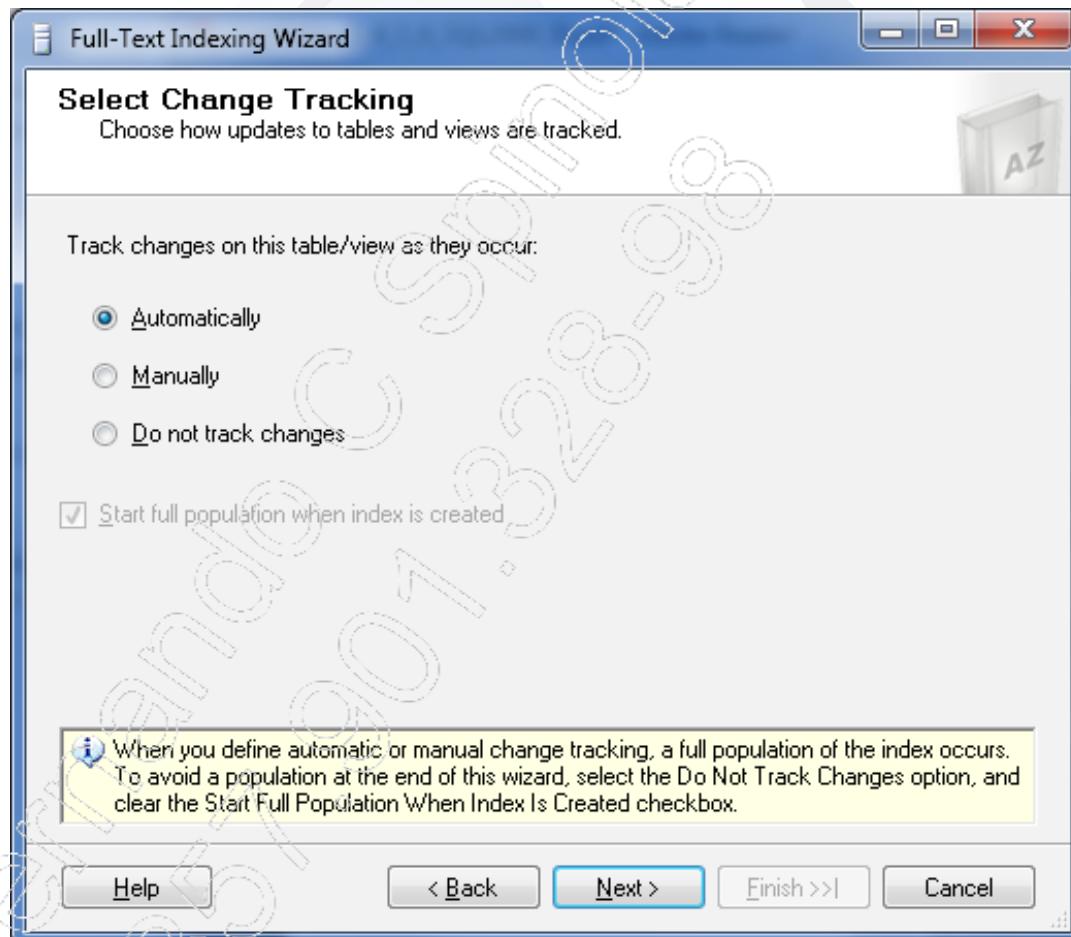


10. Clique em **Next**:

11. Na página seguinte, selecione as colunas sobre as quais o índice full-text será criado;

12. Clique em **Next**:

13. Na próxima página, escolha a forma de rastreamento das alterações das tabelas e views:

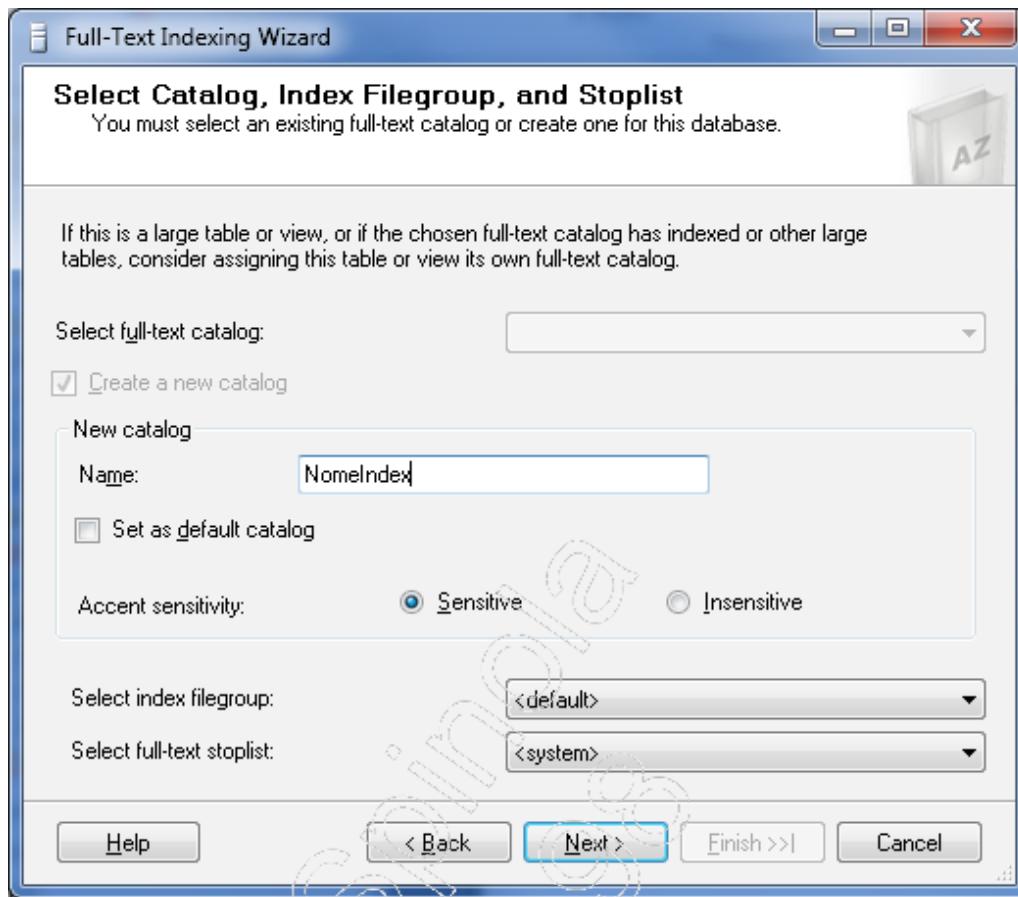


14. Clique em **Next**:

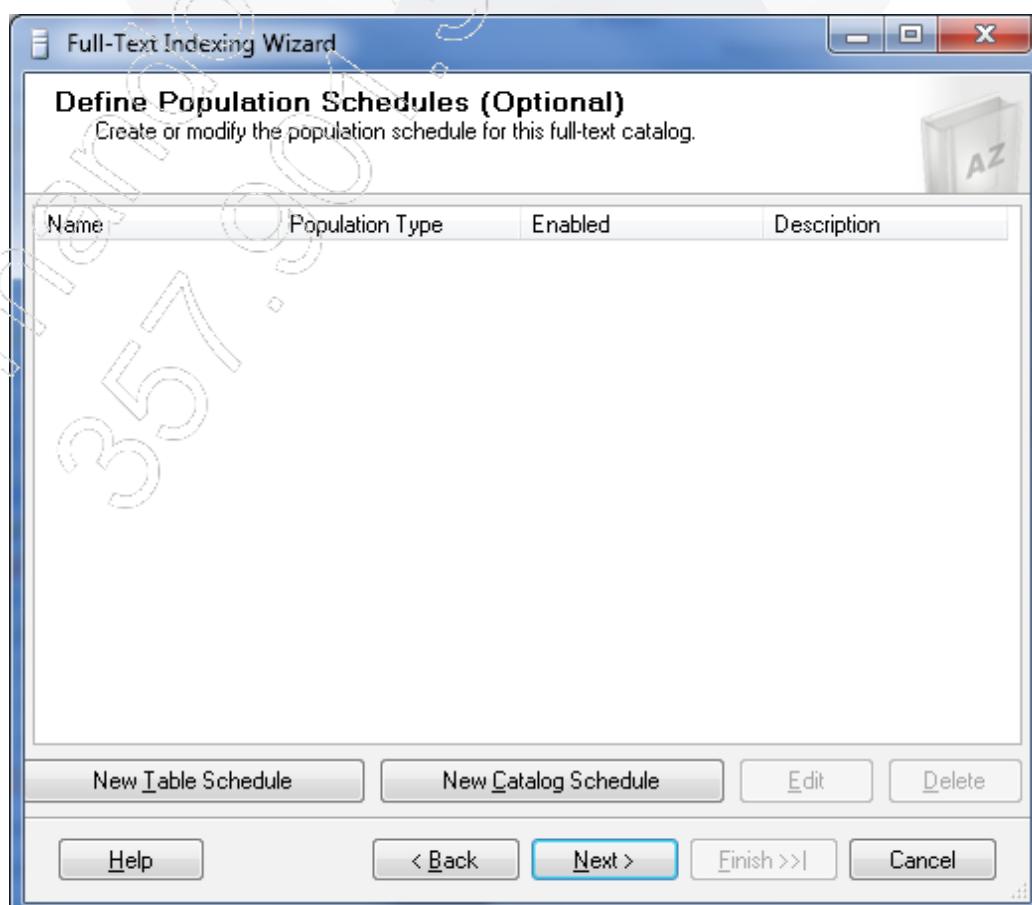
15. Na próxima página, selecione o catálogo sobre o qual o índice será criado;

SQL 2014 - Módulo III

16. Clique em **Next**:

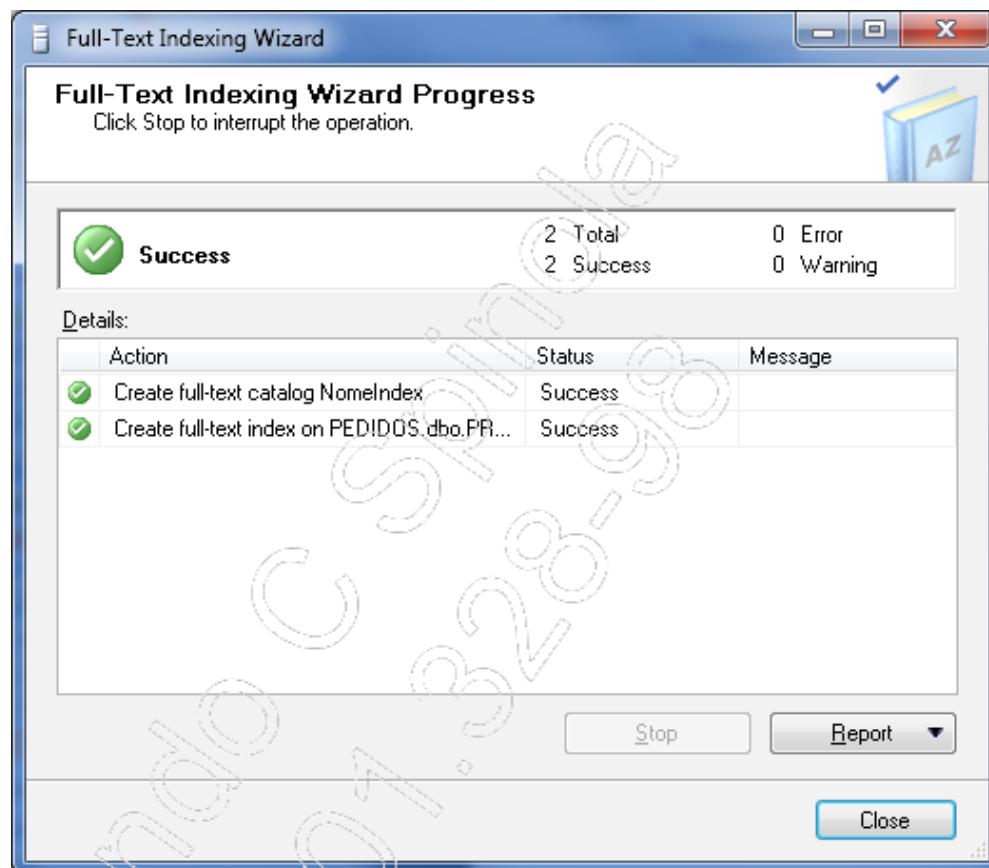


17. Configure o agendamento na página seguinte, caso haja intenção de criar um job para alimentar o índice com dados:



18. Clique em **Next**;

19. Clique em **Finish**;



20. Clique em **Close**;

21. Para popular o índice, clique com o botão direito do mouse sobre o catálogo, escolha a opção **Full-Text Index** e, em seguida, **Start Full Population**. Por fim, clique em **Close**.

4.7.1. Pesquisando em colunas FULL-TEXT

O SQL oferece funções que possibilitam consultar colunas que possuem o índice full-text. Essas consultas permitirão a utilização desse índice pelo SQL Server.

Tais funções são mais poderosas do que o comando **LIKE**, cuja forma de trabalhar é parecida.

No exemplo a seguir, é feita uma pesquisa que busca todas as linhas da tabela **Cliente** que contenham uma ou mais palavras descritas (neste caso, a palavra pesquisada é “**Atrasado**”):

```
SELECT Cod_Cli,Obs_Cli  
FROM Cliente  
WHERE FREETEXT (Obs_Cli, 'Atrasado' )
```

Já no exemplo a seguir são pesquisadas todas as linhas da tabela **Cliente** que apresentam as palavras “**Atrasado**” e “**pagou**”:

```
SELECT Cod_Cli,Obs_Cli  
FROM Cliente  
WHERE FREETEXT (Obs_Cli, 'Atrasado,pagou' )
```

No próximo exemplo, são pesquisadas as linhas da tabela **Cliente** que apresentam a palavra “**lamentável**”:

```
SELECT Cod_Cli,Obs_cli  
FROM Cliente  
WHERE CONTAINS(Obs_Cli, 'lamentável')
```

No exemplo adiante, é feita a pesquisa de todas as linhas da tabela **Cliente** que possuem a palavra “**lamentável**” ou “**muitas falhas**”:

```
SELECT Cod_Cli,Obs_cli  
FROM Cliente  
WHERE CONTAINS(Obs_Cli,' "lamentável" OR "muitas falhas"' )
```

Agora, vejamos o seguinte exemplo:

```
SELECT Cod_Cli, Obs_Cli  
FROM Cliente  
WHERE CONTAINS (Obs_Cli, ' "pag*' ')
```

No código anterior, são retornados todos os clientes cuja coluna **Obs_Cli** possua no mínimo uma palavra com o prefixo “**Pag**”.

Observemos o exemplo a seguir:

```
SELECT Cod_Cli, Obs_Cli  
FROM Cliente  
WHERE CONTAINS (Obs_Cli, 'Cometeu NEAR Ano')  
SELECT Cliente.Cod_Cli,  
B.[Key],  
Cliente.Obs_Cli  
FROM Cliente INNER JOIN CONTAINSTABLE  
(Cliente,Obs_Cli, '(Cometeu NEAR Ano)') AS B  
ON Cliente.Cod_Cli = b.[Key]  
SELECT Cliente.Cod_Cli,  
B.[Key],  
Cliente.Obs_Cli  
FROM Cliente INNER JOIN FREETEXTTABLE  
(Cliente,Obs_Cli, '(Cometeu NEAR Ano)') AS B  
ON Cliente.Cod_Cli = B.[Key]
```

No código anterior, são retornados todos os clientes que apresentam a palavra “**Cometeu**” próxima da palavra “**Ano**”.

Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo.

- Recomenda-se a criação de índices e tabelas em grupos de arquivos (FILEGROUP) diferentes, pois, com isso, tende-se a usar I/O em paralelo, aumentando a performance do banco de dados;
- Através dos índices, podemos encontrar rapidamente uma determinada informação. O local em que essa informação está armazenada é indicado através de apontadores. Um índice também possui chaves definidas com base em uma ou mais colunas de uma tabela;
- Os dados são ordenados fisicamente e armazenados na própria tabela pelo SQL Server com base nos valores de chave definidos pelo índice Clustered. No caso dos índices NonClustered, os dados são ordenados de maneira lógica. Em um índice Unique (também conhecido como único), a coluna indexada não pode conter valores repetidos;
- Para que a execução das queries atinja uma excelente performance, é preciso definir um plano de execução otimizado. A escolha do plano de execução mais apropriado pode ser feita por meio do otimizador de consultas;
- Uma hash join permite que o otimizador de consultas execute um plano de execução para uma determinada query que possui uma ou várias joins, principalmente quando as tabelas definidas pelo join não apresentam índices adequados. No entanto, uma hash join pode afetar a performance de execução de uma join. Isso acontece quando a query é utilizada diversas vezes.

4

índices Teste seus conhecimentos

Fernando C. Spinola
357.907-38-98



IMPACTA
EDITORA

1. Qual das alternativas a seguir não é um tipo de índice?

- a) Clustered
- b) NonClustered
- c) Primary Key
- d) Unique
- e) Composto

2. O que deve ser considerado para a criação de um índice?

- a) Seletividade.
- b) Densidade.
- c) Seletividade e densidade.
- d) Seletividade, densidade e estatística.
- e) Nenhuma das alternativas anteriores.

3. Como podemos sobrepor o otimizador do SQL?

- a) Com o HINT NOLOCK.
- b) Com o HINT INDEX.
- c) Não é possível sobrepor o otimizador.
- d) Através da criação de índices.
- e) Com o HINT NEW INDEX.

4. Qual a funcionalidade de índices Full-Text?

- a) Melhoria na pesquisa de campos do tipo texto.
- b) Acesso dinâmico a campos numéricos.
- c) Redução do tamanho de campos texto.
- d) Aumento de performance no SQL.
- e) Diminuição de I/O.

5. Por que o uso de índices pode melhorar a performance de um banco de dados?

- a) A utilização de índices evita FULL TABLE SCAN, diminuindo a utilização de disco.
- b) O acesso sempre será através da tabela e, depois, do índice.
- c) A utilização de disco não interfere na performance do SQL.
- d) Não há diferença, pois o SQL optimiza o acesso ao dado.
- e) É um recurso antigo que deve ser substituído por VIEW.

4

Índices Mãos à obra!

Fernando Cipinola
357.907.328-98



IMPACTA
EDITORA

Laboratório 1

A – Anexando o database Siscom (sistema comercial)

1. Crie uma pasta **SISCOM** em **C:\Bancos**;
2. Copie os arquivos **Siscom_Data.MDF** e **Siscom_Log.LDF** da pasta **C:\SQLServer2014 - Módulo III\Capitulo_04** para **C:\Bancos\SISCOM**;
3. Abra o **SQL Server Management Studio** conectando-se com a autenticação do Windows;
4. Abra e execute o **Script_01** localizado na pasta **C:\SQLServer2014 - Módulo III\Capitulo_04**;
5. Do lado esquerdo da tela, no **Object Explorer**, verifique se o database **Siscom** já faz parte da lista de databases do seu servidor. Se necessário, pressione o botão direito do mouse sobre a pasta **Databases** e escolha a opção **Refresh** para que o database seja exibido.

B – Analisando o database Siscom

1. No **SQL Server Management Studio**, do lado esquerdo da tela, no **Object Explorer**, expanda o **Database Siscom** e verifique a existência das tabelas **Cliente**, **Dependente**, **Funcionario**, **ItensPedido**, **Pedido** e **Produto**;
2. Na barra de ferramentas, clique na opção **New Query**, conectando-se com a autenticação do Windows;
3. Execute a procedure **SP_HelpIndex** em cada uma das tabelas citadas anteriormente para verificar que não há índice em nenhuma delas. Se preferir, abra o **Script_02** da pasta **Capitulo_04** para obter os comandos já escritos;

4. Agora, vamos verificar quantas linhas cada tabela possui e quantas páginas cada uma das tabelas está utilizando. Para tanto, abra o **Script_03** da pasta **Capítulo_04**, execute os comandos lá escritos e preencha a tabela a seguir:

Nome Tabela	Qtd. Linhas	Qtd. Páginas
Dependente		
Funcionario		
ItensPedido		
Pedido		
Produto		

5. Abra e execute os comandos do **Script_04** da pasta **Capítulo_04** e preencha a tabela a seguir:

Nome da tabela: Dependente	
Item analisado	Valor encontrado
Pages Scanned	
Extents Scanned	
Extent Switches	
Avg. Pages per Extent	
Scan Density [Best Count:Actual Count]	
Extent Scan Fragmentation	
Avg. Bytes Free per Page	
Avg. Page Density (full)	

SQL 2014 - Módulo III

Nome da tabela: Funcionario	
Item analisado	Valor encontrado
Pages Scanned	
Extents Scanned	
Extent Switches	
Avg. Pages per Extent	
Scan Density [Best Count:Actual Count]	
Extent Scan Fragmentation	
Avg. Bytes Free per Page	
Avg. Page Density (full)	

Nome da tabela: ItensPedido	
Item analisado	Valor encontrado
Pages Scanned	
Extents Scanned	
Extent Switches	
Avg. Pages per Extent	
Scan Density [Best Count:Actual Count]	
Extent Scan Fragmentation	
Avg. Bytes Free per Page	
Avg. Page Density (full)	

Nome da tabela: Pedido	
Item analisado	Valor encontrado
Pages Scanned	
Extents Scanned	
Extent Switches	
Avg. Pages per Extent	
Scan Density [Best Count:Actual Count]	
Extent Scan Fragmentation	
Avg. Bytes Free per Page	
Avg. Page Density (full)	

Nome da tabela: Produto	
Item analisado	Valor encontrado
Pages Scanned	
Extents Scanned	
Extent Switches	
Avg. Pages per Extent	
Scan Density [Best Count:Actual Count]	
Extent Scan Fragmentation	
Avg. Bytes Free per Page	
Avg. Page Density (full)	

6. Execute algumas queries e verifique quantas linhas cada uma delas obtém das tabelas e o tempo que cada uma leva para obter os dados. Analise o plano de execução escolhido pelo **Otimizador** para a execução de cada uma das queries. O plano de execução é visualizado marcando a consulta e pressionando **CTRL + L**. Abra o **Script_05** da pasta **Capítulo_04** e execute as queries uma a uma;

Query_01: Exibir todos os clientes que fizeram pedidos	
Item analisado	Valor encontrado
Tabelas lidas pela query	1 - 2 -
Qtd. de linhas obtidas pela query	
Tempo de execução da query	
Operadores do Plano de Execução	1 - 2 - 3 - 4 -
De que tabela o SQL Server obteve mais dados?	

SQL 2014 - Módulo III

Query_02: Exibir quantos pedidos cada cliente possui	
Item analisado	Valor encontrado
Tabelas lidas pela query	1 - 2 -
Qtd. de linhas obtidas pela query	
Tempo de execução da query	
Operadores do Plano de Execução	1 - 2 - 3 - 4 - 5 -
De que tabela o SQL Server obteve mais dados?	

Query_03: Os dependentes que cada funcionário possui	
Item analisado	Valor encontrado
Tabelas lidas pela query	1 - 2 -
Qtd. de linhas obtidas pela query	
Tempo de execução da query	
Operadores do Plano de Execução	1 - 2 - 3 - 4 - 5 - 6 -
De que tabela o SQL Server obteve mais dados?	

Query_04: Mostrar os detalhes dos pedidos de cada cliente

Item analisado	Valor encontrado
Tabelas lidas pela query	1 - 2 - 3 -
Qtd. de linhas obtidas pela query	
Tempo de execução da query	
Operadores do Plano de Execução	1 - 2 - 3 - 4 - 5 - 6 -
De que tabelas o SQL Server obteve mais dados?	1 - 2 -

Query_05: Mostrar os detalhes do pedido 4837

Item analisado	Valor encontrado
Tabelas lidas pela query	1 - 2 - 3 -
Qtd. de linhas obtidas pela query	
Tempo de execução da query	
Operadores do Plano de Execução	1 - 2 - 3 - 4 - 5 - 6 -
De que tabela o SQL Server obteve mais dados?	1 -

SQL 2014 - Módulo III

7. Execute o **Script_06** da pasta **Capítulo_04** para acrescentar as chaves primárias, as chaves estrangeiras e alguns índices nas tabelas;
8. Execute o **Script_02** (novamente) da pasta **Capítulo_04** e, com as informações obtidas, preencha a tabela a seguir:

Nome da tabela	Nome dos índices existentes na tabela	Tipo do índice
Cliente	1 -	1 -
Dependente	1 - 2 -	1 - 2 -
Funcionario	1 -	1 -
ItensPedido	1 - 2 - 3 - 4 - 5 -	1 - 2 - 3 - 4 - 5 -
Pedido	1 - 2 - 3 -	1 - 2 - 3 -
Produto	1 -	1 -

9. Execute as mesmas queries processadas no passo 6 deste laboratório, considerando que agora as tabelas já possuem índices. Verifique quantas linhas cada uma delas obtém das tabelas, o tempo que cada uma leva para obter os dados e analisar o plano de execução escolhido pelo **Otimizador** para a execução de cada uma das queries. Para isso, abra novamente o **Script_05** da pasta **Capítulo_04** e execute as queries uma a uma;

Query_01: Exibir todos os clientes que fizeram pedidos	
Item analisado	Valor encontrado
Tabelas lidas pela query	1 - 2 -
Qtd. de linhas obtidas pela query	
Tempo de execução da query	
Operadores do Plano de Execução	1 - 2 - 3 - 4 -
De que tabela o SQL Server obteve mais dados?	

SQL 2014 - Módulo III

Query_02: Exibir quantos pedidos cada cliente possui

Item analisado	Valor encontrado
Tabelas lidas pela query	1 - 2 -
Qtd. de linhas obtidas pela query	
Tempo de execução da query	
Operadores do Plano de Execução	1 - 2 - 3 - 4 - 5 -
De que tabela o SQL Server obteve mais dados?	

Query_03: Os dependentes que cada funcionário possui

Item analisado	Valor encontrado
Tabelas lidas pela query	1 - 2 -
Qtd. de linhas obtidas pela query	
Tempo de execução da query	
Operadores do Plano de Execução	1 - 2 - 3 - 4 - 5 - 6 -
De que tabela o SQL Server obteve mais dados?	

Query_04: Mostrar os detalhes dos pedidos de cada cliente

Item analisado	Valor encontrado
Tabelas lidas pela query	1 - 2 - 3 -
Qtd. de linhas obtidas pela query	
Tempo de execução da query	
Operadores do Plano de Execução	1 - 2 - 3 - 4 - 5 - 6 -
De que tabelas o SQL Server obteve mais dados?	1 - 2 -

Query_05: Mostrar os detalhes do pedido 4837

Itens analisados	Valor encontrado
Tabelas lidas pela query	1 - 2 - 3 -
Qtd. de linhas obtidas pela query	
Tempo de execução da query	
Operadores do Plano de Execução	1 - 2 - 3 - 4 - 5 - 6 -
De que tabela o SQL Server obteve mais dados?	1 -

SQL 2014 - Módulo III

10. Analise agora os resultados obtidos nos passos 6 e 9. Verifique se houve alguma melhora ao executar as queries com e sem índice. Para isso, preencha as tabelas a seguir:

Query_01: Exibir todos os clientes que fizeram pedidos	
Sem Índice	
Com Índice	

Query_02: Exibir quantos pedidos cada cliente possui	
Sem Índice	
Com Índice	

Query_03: Os dependentes que cada funcionário possui	
Sem Índice	
Com Índice	

Query_04: Mostrar os detalhes dos pedidos de cada cliente

Sem Índice	
Com Índice	

Query_05: Mostrar os detalhes do pedido 4837

Sem Índice	
Com Índice	

Gerenciando a recuperação de dados

5

- ✓ Planejando o Backup/Restore;
- ✓ Roles para execução de backup;
- ✓ Mídia para armazenar backups;
- ✓ Devices de backup;
- ✓ Atividades que não podem ser executadas durante o processo de backup;
- ✓ Modo de recuperação;
- ✓ Modalidades de backups;
- ✓ Realizando backups;
- ✓ Backup utilizando ambiente gráfico;
- ✓ Restauração de um backup;
- ✓ Restauração utilizando ambiente gráfico;
- ✓ Anexando e desanexando um banco de dados.

5.1. Introdução

O gerenciamento de cópia (backup) e de restauração (restore) são atividades importantíssimas na administração de um banco de dados SQL Server. Perdas e falhas podem ocorrer ao longo da vida de um banco de dados, preveni-las é uma tarefa essencial. Perdas relacionadas a erros causados por manipulação indevida de dados e falhas de hardware consistem nos principais motivos de recuperação de um banco de dados. Veremos, neste capítulo, as técnicas e formas de realizarmos backups, assim como realizar recuperação de dados de maneira pontual e precisa.

5.2. Planejando o Backup/Restore

Devemos considerar alguns aspectos importantes relacionados à criação do backup, a destacar:

- O backup deve ser programado para ser executado no momento em que o SQL Server não está realizando nenhum processamento complexo;
- Um sistema destinado para OLAP (tomada de decisão) não exige uma periodicidade muito frequente para a implementação do backup, assim como um sistema que não possua muitas atividades;
- Nos backups, podemos encontrar a localização dos arquivos originais. Esta localização é registrada pelo SQL Server para uma posterior utilização na recuperação e recriação dos arquivos perdidos;
- No SQL Server, o processo de backup é dinâmico. Isso significa que os usuários podem continuar utilizando as informações do banco de dados enquanto o backup é realizado;
- O backup também envolverá as atividades realizadas no momento em que ele é criado. Dessa maneira, as atividades ocorridas até o início do backup podem ser encontradas em partes do transaction log que são copiadas para o backup.

Também é necessário definir:

- Tempo de retenção do backup;
- Tempo máximo para restauração do ambiente;
- Local de armazenamento das cópias de backup;
- Usuários que realizam o backup.

5.3. Roles para execução de backup

No SQL Server, o processo de backup pode ser executado apenas pelos membros dos seguintes roles:

- **Fixed database role db_backupoperator;**
- **Fixed database role db_owner;**
- **Fixed server role sysadmin.**

5.4. Mídia para armazenar backups

Os backups realizados pelo SQL Server podem ser armazenados em dois tipos de mídia: fita (tape) ou em disco (hard disk).

- **Tape Backup Devices (dispositivos de backup em fita)**

Para que um backup seja armazenado em fita, devemos conectar este dispositivo fisicamente ao servidor no qual o SQL Server encontra-se instalado, ou seja, o dispositivo não pode estar disponível remotamente, apenas localmente.

Durante o processo de backup, o SQL Server poderá solicitar uma nova fita caso a atual não tenha espaço suficiente para armazenar todos os dados desejados. Dessa forma, basta fornecer outro fita para que a operação de backup prossiga normalmente.

- **Disk Backup Devices (dispositivos de backup em disco)**

Estes devices correspondem a arquivos do sistema operacional que podem ser configurados tanto em um disco de um servidor remoto que esteja compartilhado na rede, como em um disco pertencente ao servidor local.

Caso o processo de backup seja feito em um servidor remoto ou em um disco compartilhado na rede, a localização do arquivo deve ser especificada por meio do formato `\Servername\Sharename\Path\File`, baseado na convenção adotada pela UNC (Universal Naming Conventional). Caso o drive seja local, basta indicar sua localização de modo que o arquivo seja encontrado.

É importante salientar que o disco remoto também exige a concessão de permissões para a leitura e a gravação de arquivos, assim como acontece com o disco rígido local.

5.5. Devices de backup

O local de armazenamento de um backup realizado pelo SQL Server é chamado de device de backup. Além de ter um nome físico e um tipo de mídia, um device de backup também poderá contar com um nome lógico. Tanto os devices que possuem nomes lógicos como os que possuem nomes físicos podem ser utilizados para a execução do backup.

- **Nome físico do device de backup**

O nome físico corresponde ao nome que é conhecido pelo sistema operacional, pois se trata da identificação recebida pelo arquivo de backup no momento em que ele é criado. Quando um device de backup possui somente o nome físico, sem o nome lógico, dizemos que ele foi criado apenas fisicamente. Ao contrário do nome lógico, o nome físico não exige que o device de backup seja criado antes da execução do backup.

Vejamos a sintaxe utilizada:

```
BACKUP DATABASE Impacta  
TO Disk='C:\Backup\Backup_Impacta.bak'
```

- **Nome lógico do device de backup**

A identificação de um arquivo de backup é realizada através de um alias, mais conhecido como nome lógico. Este alias permanece arquivado para sempre nas views do sistema. É importante ressaltar que um nome lógico não pode ser definido sem que seja criado um device de backup antes da execução do próprio backup.

A system stored procedure **sp_addumpdevice** pode ser utilizada para a definição dos devices de backup, assim como a ferramenta gráfica SQL Server Management Studio. Uma vez criados logicamente, esses devices são registrados na view **sys.backup_devices**, que pode ser encontrada no banco de dados master.

O device lógico de backup permite que a sintaxe do comando seja escrita de uma maneira mais simples. Para compreendermos melhor sua utilização, consideremos os procedimentos a seguir:

1. Para criar o device, devemos utilizar o comando a seguir:

```
Exec SP_addumpdevice  
@devtype='Disk',  
@logicalname='Backup_Impacta',  
@physicalname='C:\Backup\Backup_Impacta.bak'
```

2. Para executar o backup:

```
BACKUP DATABASE Impacta  
TO Backup_Impacta
```

5.5.1. Backup set, media set, media family, initial media, continuation media

Vejamos alguns conceitos importantes a respeito das mídias utilizadas no processo de backup antes de iniciarmos sua execução:

- **Backup set:** O conteúdo de uma só operação de backup recebe essa denominação;
- **Media family:** Todas as mídias destinadas ao armazenamento de um só backup set através de um único device de backup compõem uma **media family**;
- **Media set:** O conteúdo de uma operação de backup, ou seja, o backup set, é armazenado em um conjunto de mídias chamado **media set**. Independentemente do número de devices e de mídias utilizado nesse processo, um ou vários backup sets podem ser armazenados em um **media set**;
- **Initial media:** Entre as mídias que compõem a **media family**, a **initial media** será a primeira a ser utilizada. Caso essa mídia seja completada durante o backup, poderemos utilizar outras mídias até a finalização do processo;
- **Continuation media:** Utilizadas apenas em tapes, **continuation media** são as mídias restantes necessárias para que o backup prossiga normalmente assim que a **initial media** é completada. Cada mídia física pertencente a uma **media family** possui um número que especifica sua ordem de utilização. Por exemplo, 1 refere-se à primeira mídia utilizada, 2 à segunda, e assim por diante. Dessa forma, é possível garantir o número e a ordem correta de backups realizados.

5.5.2. Usando múltiplos backup devices

Com relação à utilização de múltiplos devices de backup, é importante considerar os seguintes aspectos:

- Uma operação individual de backup permite o uso de até 64 devices de backup. Esses devices devem ser do mesmo tipo, ou seja, todos devem ser tapes ou discos;
- Caso as operações de backup sejam baseadas em múltiplos backup devices, devemos reservar a mídia utilizada somente para operações de backup realizadas pelo SQL Server;
- Assim que uma operação de backup cria um media set inteiro, todas as próximas operações deverão utilizá-lo. Suponhamos que dois devices de backup foram utilizados para a criação do media set. Dessa forma, os mesmos dois devices devem ser empregados nos backups posteriores que serão realizados para o mesmo media set;
- Ao executar operações de restore ou de backup, é possível fazer uso de múltiplos devices de backup ao mesmo tempo. Dessa forma, é possível otimizar a velocidade de ambas as operações com a utilização de I/O paralelo.

5.6. Atividades que não podem ser executadas durante o processo de backup

No SQL Server, o backup pode ser realizado enquanto o banco de dados está em plena atividade, portanto, trata-se de um backup dinâmico. Em contrapartida, algumas atividades não podem ser realizadas durante a execução do backup, especificamente:

- Criação de índices;
- Alteração ou criação de bancos de dados;
- Execução de operações não registradas no transaction log;
- Autocrescimento.

Se uma dessas atividades estiver em andamento no momento em que decidirmos iniciar o backup, este não poderá ser iniciado. Se o usuário tentar executar uma dessas atividades durante o processo de backup, o SQL Server não permitirá sua execução.

5.7. Modo de recuperação

O modo de recuperação (**RECOVERY MODEL**) é uma propriedade definida para cada banco de dados e consiste em três modos distintos:

- **FULL**;
- **BULKED-LOGGED**;
- **SIMPLE**.

Cada um desses modos têm finalidades e usos diferentes, logo a escolha de cada um deles depende dos objetivos que pretendemos alcançar.

5.7.1. Modelo de recuperação completo (FULL)

Neste modelo é possível a restauração a partir de qualquer momento até o último backup de LOG. Recomendado para bancos de dados críticos e que necessitam de recuperações pontuais. Este modo é compatível com qualquer estratégia de backup e de recuperação de dados. Deve ser usado em sistemas transacionais (OLTP).

Caso o banco de dados esteja em outro modelo de recuperação, este poderá ser alterado para o modelo completo. No entanto, após a modificação desse modelo, deve-se ter o cuidado de realizar um backup completo do banco de dados.

A seguir, temos a sintaxe para a modificação do modelo de recuperação completo:

```
ALTER DATABASE nome_db  
SET RECOVERY FULL [WITH NO_WAIT]
```

Nesse modelo, é necessário que seja feito o backup regular do arquivo de log, pois, em caso de recuperação do banco de dados, o que irá garantir a plena recuperação será o backup do arquivo de log.

O administrador deverá, ao usar esse modelo, realizar backups regulares do arquivo de log (a cada quatro horas, por exemplo). Depois de realizado o backup de log, é necessário realizar o procedimento de compactação do arquivo de log para que esse arquivo volte a um tamanho menor e possa crescer novamente até a realização do próximo backup de log.

```
--Realize o Backup FULL  
--Realize o backup de LOG  
  
ALTER DATABASE [nome_db] SET RECOVERY SIMPLE WITH NO_WAIT  
DBCC SHRINKFILE(nome_arq_log, 1)  
ALTER DATABASE [nome_db] SET RECOVERY FULL WITH NO_WAIT  
GO
```

O nome e o tamanho (em MB) do arquivo de log deverão ser informados para que o comando **DBCC SHRINKFILE** realize a redução desse arquivo.

Vejamos o procedimento a ser seguido neste modelo de recuperação:

1. Realizar o backup completo;
2. Realizar o backup de log regular;
3. Realizar a compactação do arquivo de log após o backup de log.



Lembre-se que o próximo backup deve ser FULL.

5.7.2. Modelo de recuperação BULKED-LOGGED

Este modelo de recuperação **BULKED-LOGGED** é útil quando o banco de dados realiza movimentação de dados apenas através de carga de dados (sem uso de sistemas ou aplicações cadastrais). Este tipo de recuperação é indicado para aplicações que realizam carga de dados usando o método **BULK**.

Neste modelo o LOG é utilizado de forma minimizada e o dado pode não ser consistente em operações em bloco.

Caso o banco de dados utilize outro modelo de recuperação e precisemos alterá-lo para **BULKED-LOGGED**, é necessária a realização de um backup completo após a alteração do modelo de recuperação.

A seguir, a sintaxe para a modificação do modelo de recuperação **BULKED-LOGGED**:

```
ALTER DATABASE nome_db  
SET RECOVERY BULKED_LOGGED [WITH NO_WAIT]
```

Como a maioria dos comandos não é registrada no arquivo de log, esse modelo tende a um crescimento menor, necessitando, portanto, de menos backups regulares do arquivo de log. Mesmo assim, nesse modelo é necessário o backup dos arquivos de log. Isso não garantirá, porém, a recuperação completa do banco de dados em caso de perda.

5.7.3. Modelo de recuperação simples (SIMPLE)

Este modelo de recuperação simples é útil quando o banco de dados não necessita de recuperação até o momento em que a falha ocorreu. Basta a simples recuperação do backup e as alterações realizadas após este último serão perdidas.

Ele provocará necessariamente perdas de dados. Para minimizar a quantidade de dados perdidos, é recomendado o backup dos dados de forma mais regular que no modo de recuperação completo.

Neste modelo, os dados do arquivo de log são compactados logo após a aplicação da transação, minimizando assim a ocupação do arquivo de log em disco. Neste caso, não há a necessidade de backup de log e nem a compactação deste arquivo.

O modelo **SIMPLE** não é recomendado para sistemas transacionais que precisam de recuperação de dados completa.

Alguns bancos de dados de sistema usam este modelo de recuperação. Mesmo que alteremos o modelo de recuperação destes bancos de dados, ainda assim será aplicado o modelo de recuperação **SIMPLE**.

Vejamos a sintaxe para a modificação do modelo de recuperação **SIMPLE**:

```
ALTER DATABASE nome_db  
SET RECOVERY SIMPLE [WITH NO_WAIT]
```

5.8. Modalidades de backups

O SQL Server suporta duas modalidades de backups, o físico e o lógico. Cada uma tem características e utilizações próprias. O backup lógico consiste na exportação dos dados de uma ou mais tabelas para um ou mais arquivos físicos. Dessa forma se, durante o processo de exportação, os dados forem modificados, não haverá nenhuma consistência por parte do backup.

Já o backup físico é um procedimento que leva em conta a cópia física dos arquivos de dados e log. Ele possui uma consistência, pois, ao restaurar o backup dos dados e do log, o banco de dados aplicará todas as alterações contidas no arquivo de log nos arquivos de dados. Dessa forma, esse banco ficará atualizado até a última transação registrada nos arquivos de log, ou mesmo até algum ponto entre o último backup e o ponto em que ocorreu a falha no banco de dados.

O backup lógico será abordado mais adiante. Vamos tratar agora do backup físico, que pode ser feito por meio de dois procedimentos distintos:

- **Backup Frio (Cold Backup)**: Consiste em um backup do banco de dados com a base sem nenhum acesso. Para essa ação, o banco de dados deve ser colocado no modo **OFFLINE** e ter copiados os arquivos de dados e Log. Neste tipo de backup, o que se busca recuperar é o exato momento em que ele foi realizado. Alterações feitas após o término do backup serão perdidas;
- **Backup Quente (Hot Backup)**: Consiste em um backup do banco de dados e mais backups parciais do arquivo de log. Este backup ocorre em paralelo com a utilização das bases de dados, não necessitando de parada do ambiente.

5.8.1. Backup físico frio

O backup físico frio, também chamado, conforme visto anteriormente, de **cold backup**, é um backup que podemos considerar como consistente, pois deve ser realizado com o banco de dados parado.

Para paramos um banco de dados, é necessário este comando:

```
ALTER DATABASE { database_name | CURRENT }
SET OFFLINE WITH ROLLBACK IMMEDIATE
```

Esse comando fará o banco de dados ser encerrado. As transações em andamento que não receberam **COMMIT** serão desfeitas e o banco de dados ficará em estado íntegro para cópia.

Após a execução do comando, os arquivos de dados e log deverão ser copiados para um local ou mídia apropriada (Fita, Disco ou DVD/Blu-Ray).

Depois que os arquivos forem copiados, o banco de dados precisa voltar a ficar on-line, o que pode ser feito através do seguinte comando:

```
ALTER DATABASE { database_name | CURRENT }
SET ONLINE
```

A responsabilidade pela cópia é integral do processo de backup, não havendo nenhuma ferramenta ou recurso do SQL Server envolvido, exceto se a tarefa tiver sido agendada através do SQL Agent.

Para restaurar este tipo de backup, é necessário parar o banco de dados, restaurar os arquivos que foram copiados para o local original dos arquivos de dados e log, sobrepondo estes últimos, e realizar a abertura do banco de dados.

5.8.2. Backup físico quente

Este tipo de backup, também conhecido, conforme visto anteriormente, como **hot backup**, pode ser realizado sem que o banco de dados precise ser paralisado como no backup frio. É uma técnica mais utilizada de backup, pois não requer nenhum tipo de parada do banco de dados.

Este tipo de backup é conhecido como inconsistente, pois, para que o backup seja recuperado sem perda de dados, é necessário que o backup dos dados seja restaurado e, em seguida, todos os backups de log sejam restaurados em sequência, fazendo com que os arquivos de dados do backup inicial sejam atualizados com a aplicação de todos os backups de logs realizados.

Neste tipo de backup, é possível não termos nenhuma perda, desde que sejam aplicados todos os backups de log até o momento em que a falha ocorreu.

Também é possível realizarmos uma operação chamada de restauração em um ponto específico no tempo (**POINT IN-TIME RECOVERY**), por meio da qual a recuperação do banco de dados ocorre até um momento no tempo anterior ao momento atual do banco de dados. Por exemplo, um backup completo foi realizado à meia-noite (0h) e backups de logs são realizados a cada 1 hora. Vamos supor que, às 17h, devemos restaurar o backup realizado à meia-noite, porém, o banco de dados precisa ser restaurado às 14h35. Com isso, os logs até as 15h seriam recuperados, sendo que o último seria recuperado parcialmente (35 minutos de transações).

5.9. Realizando backups

O backup de um banco de dados pode ser realizado das seguintes formas:

- **Backup completo;**
- **Backup incremental;**
- **Backup de log;**
- **Backup de filegroup;**
- **Backup de tail log.**

Toda estratégia de backup de um banco de dados deve iniciar por um backup completo. A partir deste, temos várias opções para realização do procedimento de backup.

5.9.1. Backup completo

O backup completo deve ser o primeiro componente da rotina de backup de um banco de dados. Neste tipo de backup, todos os arquivos e dados são incorporados (dados e log).

Bancos de dados pequenos e de médio volume (até 2GB) devem ser tratados apenas por backups completos.

Vejamos a sintaxe para um comando de backup completo ou diferencial:

```
BACKUP DATABASE { nome_db | @nome_db_var }
    TO <backup_device> [ ,...n ]
    [ <MIRROR TO clause> ] [ next-mirror-to ]
    [ WITH { DIFFERENTIAL | <general_WITH_options> [ ,...n ] } ]
[ ; ]
```

SQL 2014 - Módulo III

- Argumentos do comando:

```
<backup_device>::=
{
    { logical_device_name | @logical_device_name_var }
| { DISK | TAPE } =
    { 'physical_device_name' | @physical_device_name_var }
}

<MIRROR TO clause>::=
MIRROR TO <backup_device> [ ,...n ]

<file_or_filegroup>::=
{
    FILE = { logical_file_name | @logical_file_name_var }
| FILEGROUP = { logical_filegroup_name | @logical_filegroup_name_var }
}
}

<read_only_filegroup>::=
FILEGROUP = { logical_filegroup_name | @logical_filegroup_name_var }

<general_WITH_options> [ ,...n ]::=
--Opções de backup SET (Conjuntos de backups)
COPY ONLY
| { COMPRESSION | NO_COMPRESSION }
| DESCRIPTION = { 'text' | @text_variable }
| NAME = { backup_set_name | @backup_set_name_var }
| { EXPIREDATE = { 'date' | @date_var }
    | RETAINDAYS = { days | @days_var } }

--Opções de Media SET (Conjunto de mídias de backup)
{ NOINIT | INIT }
| { NOSKIP | SKIP }
| { NOFORMAT | FORMAT }
| MEDIADESCRIPTION = { 'text' | @text_variable }
| MEDIANAME = { media_name | @media_name_variable }
| BLOCKSIZE = { blocksize | @blocksize_variable }

--Opção para transferência de dados
BUFFERCOUNT = { buffercount | @buffercount_variable }
| MAXTRANSFERSIZE = { maxtransfersize | @maxtransfersize_variable }

--Opções para gerenciamento de Erros
{ NO_CHECKSUM | CHECKSUM }
| { STOP_ON_ERROR | CONTINUE_AFTER_ERROR }

--Opções de Compatibilidade
RESTART

--Opções de Monitoração
STATS [ = percentage ]

--Opções de backup em fita
{ REWIND | NOREWIND }
| { UNLOAD | NOUNLOAD }
```

- Sintaxe para backup de filegroup:

```
BACKUP DATABASE { database_name | @database_name_var }
    <file_or_filegroup> [ ,...n ]
    TO <backup_device> [ ,...n ]
    [ <MIRROR TO clause> ] [ next-mirror-to ]
    [ WITH { DIFFERENTIAL | <general_WITH_options> [ ,...n ] } ]
[;]
```

- Sintaxe para backup de log:

```
BACKUP LOG { database_name | @database_name_var }
    TO <backup_device> [ ,...n ]
    [ <MIRROR TO clause> ] [ next-mirror-to ]
    [ WITH { <general_WITH_options> | <log-specific_optionspec> } ]
[ ,...n ] ]
[;]
```

- Sintaxe para backup parcial:

```
BACKUP DATABASE { database_name | @database_name_var }
    READ_WRITE_FILEGROUPS [ , <read_only_filegroup> [ ,...n ] ]
    TO <backup_device> [ ,...n ]
    [ <MIRROR TO clause> ] [ next-mirror-to ]
    [ WITH { DIFFERENTIAL | <general_WITH_options> [ ,...n ] } ]
[;]
```

- Argumentos do backup:

- **DATABASE**: Refere-se ao backup do banco de dados (dados e log);
- **Nome_db|@Nome_db_var**: Indica o nome do banco de dados que será copiado, caso seja informada uma variável (@**nome_db_var**);
- **Backup_device**: Tipo de dispositivo que será utilizado durante o procedimento de backup;
- **{DISK|TAPE}={‘physical_device_name’}@physical_device_name_var}**: Especifica se os backups serão realizados em fita ou disco;

- **MIRROR TO:** Especifica se os backups deverão ser feitos em mais de um local (espelhamento de backup). A cláusula **TO** deverá ser utilizada para definir o primeiro dispositivo que realizará a cópia, e a cláusula **MIRROR TO**, para definir os dispositivos do espelhamento;
- **BLOCKSIZE={BLOCKSIZE}@BLOCKSIZE_VARIABLE:** Por meio desta cláusula, a dimensão do bloco utilizado para o procedimento de backup é indicada em bytes. Ao usar essa cláusula com o tipo de backup fita (**TAPE**), não estará disponível a opção de escolha automática do tamanho do bloco. Esta opção não é utilizada para backups realizados em disco (**DISK**);
- **CHECKSUM:** Esta opção faz o backup verificar as páginas antes de salvá-la na mídia de backup. Ela é armazenada na respectiva mídia e serve para validar se o backup apresenta ou não algum tipo de corrupção. Recomendado para backup em fita (**TAPE**);
- **NO_CHECKSUM:** Cláusula padrão que desativa o processo de checagem de backup;
- **STOP_ON_ERROR:** Cláusula que interrompe a execução de um backup no caso de encontrar algum erro ou falha, impedindo que um backup com problemas seja realizado;
- **CONTINUE_AFTER_ERROR:** Indica que o backup deverá prosseguir mesmo após a detecção de algum tipo de erro. Caso um banco de dados apresente erro, esse argumento é recomendado para cópia dos arquivos de log;
- **DESCRIPTION={text}@text_variable:** Descrição do backup realizado. Este texto deve conter no máximo 255 caracteres;
- **DIFFERENTIAL:** Por meio desta opção, podemos especificar que o backup do banco de dados abrangerá apenas as alterações feitas desde o último backup completo ou diferencial, o que tiver sido feito anteriormente, ocupando um espaço menor que o backup completo;
- **EXPIREDATE={date}@var_date:** Data de expiração do conjunto de backup (BACKUP SET). Após essa data, o backup poderá ser sobreescrito. Ao utilizar esta opção, não deve ser utilizada a opção **RETAINDEAYS**;

- **RETAINDAYS={days}@var_days}**: Quantidade de dias para que o backup seja mantido. Após esse número de dias, o backup deverá ser sobreescrito. Ao escolher esta opção, não se deve utilizar **EXPIREDATE**;
- **PASSWORD={password}@var_password**: Esta opção deverá ser utilizada para proteção do conjunto de backup, para realizar a restauração, será necessária a indicação da senha;
- **FORMAT**: Indica que o conjunto de backup deve ser invalidado e o cabeçalho da mídia será reescrito;
- **NO_FORMAT**: Opção padrão que não sobrescreve o cabeçalho de mídia, exceto se for utilizada em conjunto com a opção **INIT**;
- **INIT**: Esta opção indica que o conjunto de backup deve ser reiniciado. Dessa forma, os dados serão sobrepostos;
- **NO_INIT**: Opção padrão que indica que o backup incluirá um backup set;
- **MEDIADESCRIPTION={text}@var_text**: Texto que indica as características da mídia utilizada por meio de texto com até 255 caracteres;
- **MEDIANAME={media_name}@var_media_name**: Nome do conjunto de backup com até 128 caracteres, devendo conter o mesmo nome da mídia definida anteriormente. Se não houve especificação desse nome ou se a cláusula SKIP foi usada, não será feita verificação desse nome;
- **MEDIAPASSWORD={mediapassword}@var_media_password**: Senha especificada para o conjunto de mídias de backup. Essa senha será solicitada antes da recuperação e da criação do conjunto de backups;
- **NAME={backup_set_name}@var_backup_set_name**: Nome do conjunto de backups com até 128 caracteres. Caso o nome não seja informado, ficará em branco;

- **NORECOVERY**: Esta cláusula deverá ser utilizada quando a restauração do backup incluir também a restauração dos logs, recomendado para bancos de dados que tenham como modelos de recuperação (**FULL** e **BULKED_LOGGED**);
- **SKIP**: Desativa a verificação do nome do conjunto de backup (**BACKUP SET**) e de sua data de expiração. Utilizando esta opção, é possível evitar a sobreposição dos conjuntos de backup;
- **NOSKIP**: Realiza a validação do nome do conjunto de backup (**BACKUP SET**) e de sua data de expiração. Essa operação ocorrerá antes que esse conjunto seja sobrescrito;
- **REWIND**: Opção padrão. Rebobina a fita e, ao final, ejeta-a;
- **NOREWIND**: Mantém a fita na posição final de gravação;
- **NOUNLOAD**: Não ejeta a fita ao final do backup;
- **UNLOAD**: Opção padrão desde que não seja especificada a cláusula **NOUNLOAD**. Após o backup, a fita deve ser rebobinada e ejetada do dispositivo de backup;
- **RESTART**: Permite o reinício de uma operação de backup previamente interrompida. Essa opção é válida para backups realizados em vários conjuntos de mídias (**MIRROR**);
- **STATS=[PERCENTAGE]**: Assim que o backup atinge um percentual de conclusão, uma mensagem será exibida. Caso não seja definida esta cláusula, o percentual de 10% será considerado;
- **COPY_ONLY**: Esta opção não interfere na política de backup realizada, indicando apenas que um backup está sendo realizado, não evitando os demais backups regulares. Só é possível utilizá-lo para os modelos de recuperação **FULL** e **BULKED LOGGED**. Neste caso, o arquivo de log não deverá ser truncado. Não pode ser usado em bancos de dados com estratégia diferencial;

- **FILE={logical_file_name}@var_logical_file_name}**: Podemos especificar um ou mais arquivos que serão incluídos no backup do banco de dados;
- **LOG**: Indica que apenas o backup de log deve ser realizado. A partir do fim deste backup, as transações que não estiverem ativas poderão ser eliminadas através da compactação do arquivo de log;
- **NO_TRUNCATE**: Opção que deve ser utilizada quando o banco de dados está danificado, permitindo a execução do backup do log.

5.9.2. Backup de log

O backup de log é o procedimento utilizado na estratégia de backup para recuperação do banco de dados e consiste na cópia das transações do arquivo de log que não foram copiadas desde o último backup completo ou de log. Fundamental para o uso no modelo de recuperação completa (**FULL**).

A sintaxe para backup de log é:

```
BACKUP LOG { database_name | @database_name_var }
    TO <backup_device> [ ,...n ]
    [ <MIRROR TO clause> ] [ next-mirror-to ]
    [ WITH { <general_WITH_options> | <log-specific_optionspec> } [
        ,...n ] ]
    [ ; ]
```

5.9.3. Backup parcial de arquivo e de grupo de arquivo

O backup de grupo de arquivo e o backup de arquivo são procedimentos utilizados para a realização de backups apenas de partes de um banco de dados. Isso tende a tornar a restauração mais específica para determinados arquivos e ou grupos de arquivos. Essa opção é útil quando há um número de grupo de arquivos maior que um, pois, caso haja apenas um, essa opção não terá muito sentido. Podemos ainda realizar backup de apenas um arquivo de dados.

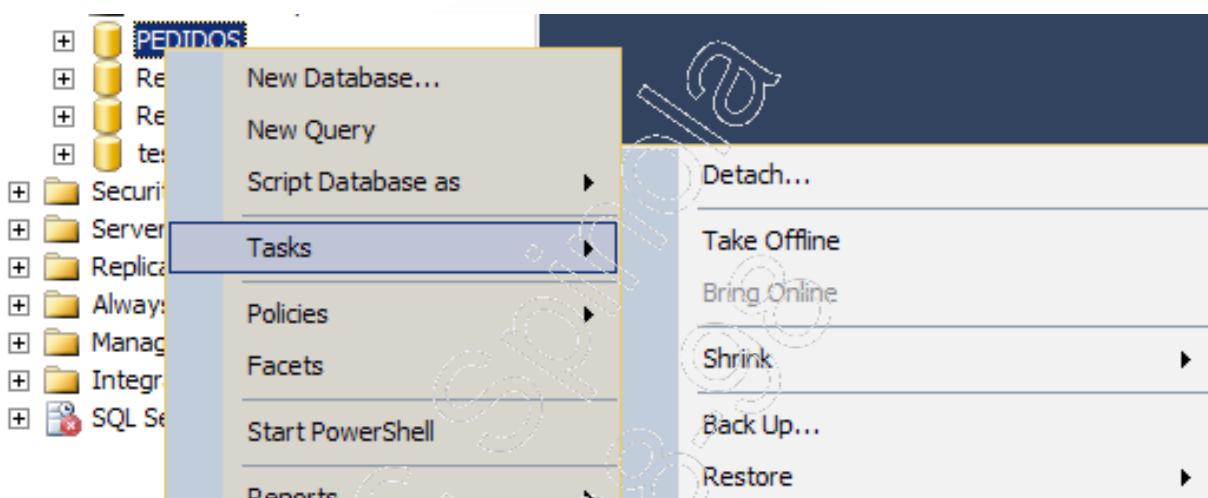
A seguir, temos a sintaxe para backup parcial:

```
BACKUP DATABASE { database_name | @database_name_var }
    READ_WRITE_FILEGROUPS [ , <read_only_filegroup> [ ,...n ] ]
    TO <backup_device> [ ,...n ]
    [ <MIRROR TO clause> ] [ next-mirror-to ]
    [ WITH { DIFFERENTIAL | <general_WITH_options> [ ,...n ] } ]
[ ; ]
```

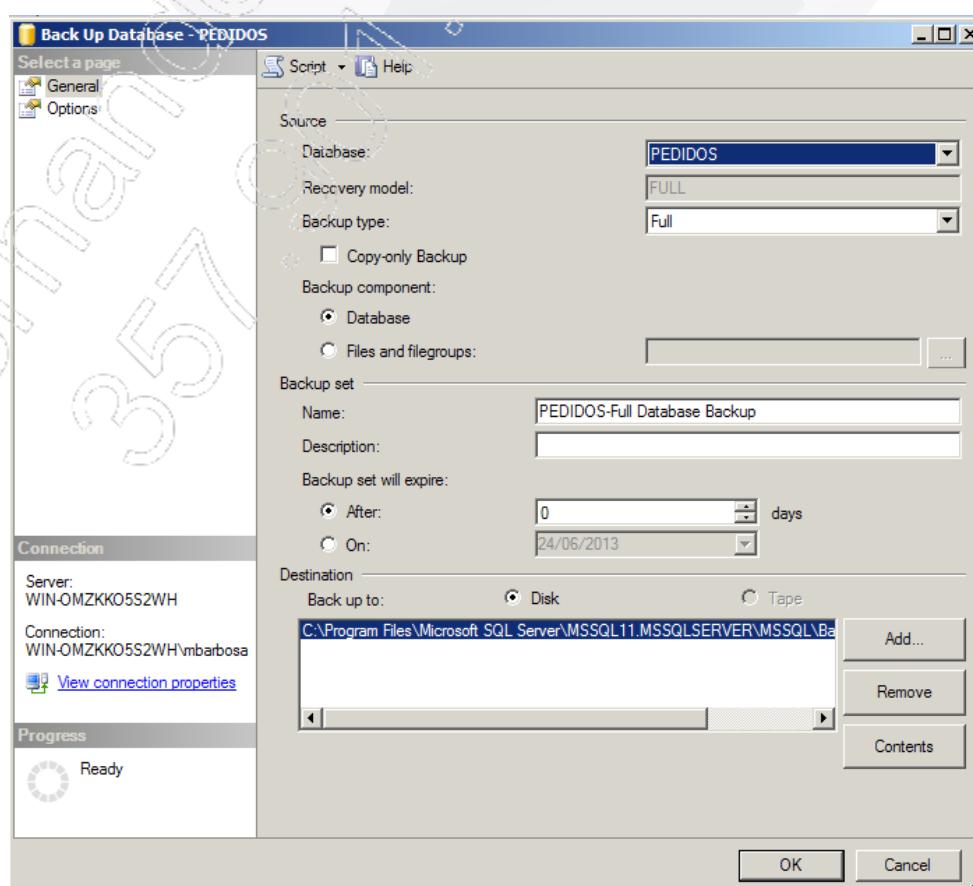
5.10. Backup utilizando ambiente gráfico

O backup pode ser comandado também através da interface gráfica do SSMS. Isso pode ser prático para execução de backups não previstos ou esporádicos. Ao final do procedimento, pode-se gerar um script SQL para a execução do backup em modo texto.

Para executarmos o backup em ambiente gráfico, é necessário posicionarmos o mouse no banco de dados que desejamos realizar o backup e selecionarmos o menu **BACKUP**. O assistente de backup abrirá a tela para seleção das opções de backup.

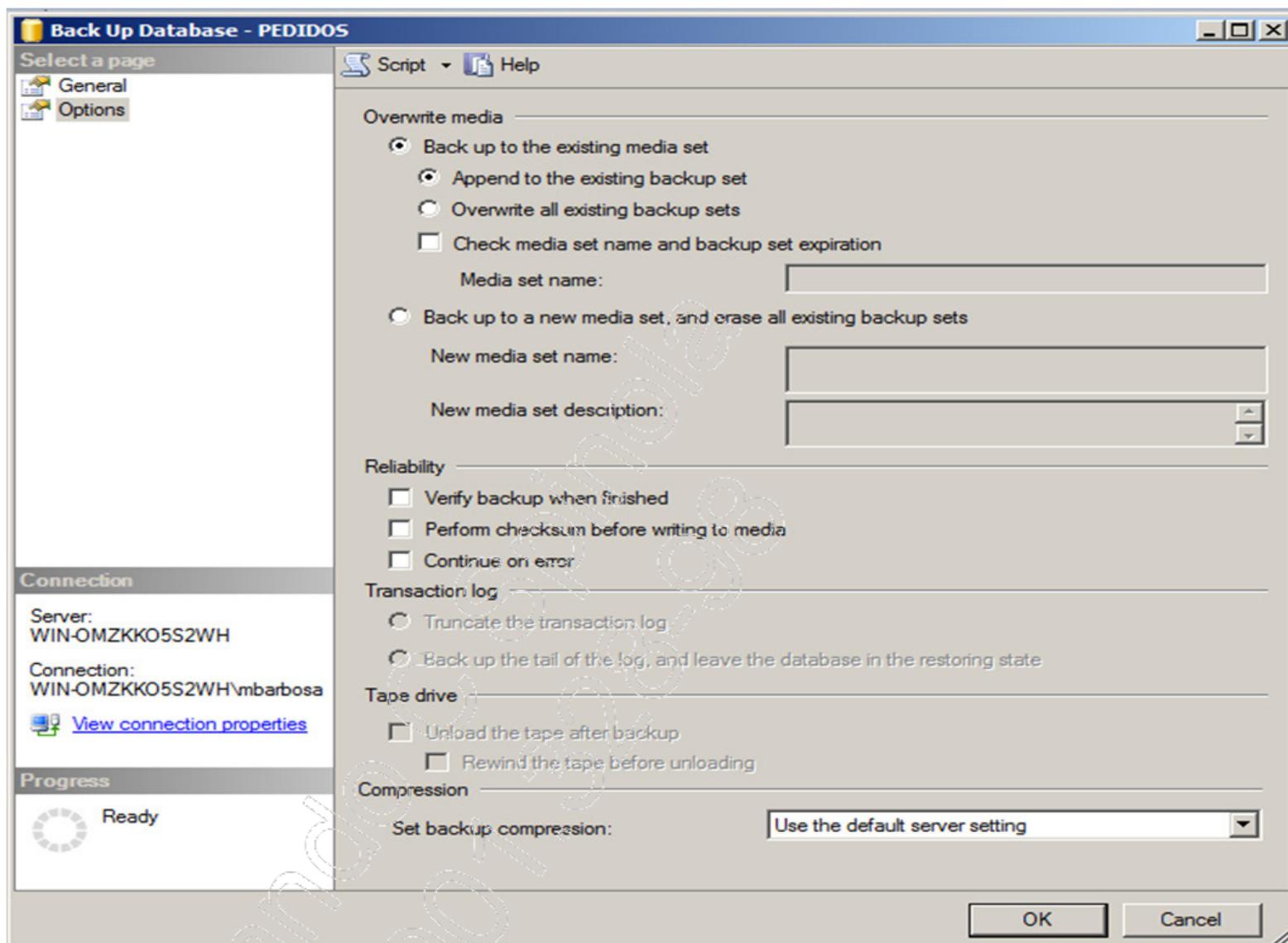


Ao selecionar a opção **Back Up**, será exibida a seguinte janela:



SQL 2014 - Módulo III

Selecione o tipo do backup (**Backup type**) e o **Backup component**. Em seguida, abra a opção **Options** e selecione as opções avançadas de backup conforme a imagem a seguir:



Selecione o botão **Script** e gere o comando com base nas opções selecionadas:



```
BACKUP DATABASE [nome_db] TO DISK = N'local físico do arquivo de
backup' WITH NOFORMAT, NOINIT, NAME = N'nome do backup', SKIP,
NOREWIND, NOUNLOAD, STATS = 10
GO
```

5.11. Restauração de um backup

A restauração de um backup depende diretamente de como o backup restaurado foi realizado. Backups frios precisam ser restaurados manualmente e o banco de dados deve estar fora de atividade (encerrado). Este tipo de backup envolve um procedimento totalmente manual, controlado apenas pelo responsável pela restauração.

Já os backups quentes devem ser restaurados através dos comandos de **RESTORE** e têm vinculação com o tipo de backup que foi realizado (Completo, Log, Filegroup ou Diferencial).

O comando **RESTORE** é o responsável pela restauração de um banco de dados, porém, é importante que ninguém esteja utilizando o banco de dados no momento da restauração.

A restauração de um banco de dados pode ser feita de acordo com as seguintes opções:

- Restauração completa de banco de dados;
- Restauração de grupo de arquivo ou arquivo de dados;
- Restauração de log;
- Restauração de página.

Vamos conhecer mais detalhadamente essas opções nos próximos subtópicos.

5.11.1.Restauração completa de banco de dados

A restauração completa de banco de dados ocorre quando um backup completo é restaurado. Os backups de log também são restaurados até o momento da falha. Neste modelo de recuperação, não há perdas. Para isso, o banco de dados deve usar o modelo de recuperação completo (**FULL**), ter um backup completo e os backups de log e, por fim, o backup de **tail log**, para garantir que a última parte do log sofreu backup.

```
RESTORE DATABASE { database_name | @database_name_var }
    <files_or_filegroups> [ ,...n ]
    [ FROM <backup_device> [ ,...n ] ]
    WITH
        PARTIAL, NORECOVERY
        [ , <general_WITH_options> [ ,...n ]
        | , <point_in_time_WITH_options-RESTORE_DATABASE>
        ] [ ,...n ]
    [ ; ]
```

Vejamos os argumentos do comando **Restore**:

```
<backup_device>::=
{
    { logical_backup_device_name |
        @logical_backup_device_name_var }
    | { DISK | TAPE } = { 'physical_backup_device_name' |
        @physical_backup_device_name_var }
}

<files_or_filegroups>::=
{
    FILE = { logical_file_name_in_backup | @logical_file_name_in_
    backup_var }
    | FILEGROUP = { logical_filegroup_name | @logical_filegroup_
    name_var }
    | READ_WRITE_FILEGROUPS
}
```

```
<general_WITH_options> [ ,...n ] ::=  
--Restore Operation Options  
    MOVE 'logical_file_name_in_backup' TO 'operating_system_file_  
name'  
        [ ,...n ]  
    | REPLACE  
    | RESTART  
    | RESTRICTED_USER  
  
--Backup Set Options  
    | FILE = { backup_set_file_number | @backup_set_file_number }  
    | PASSWORD = { password | @password_variable }  
  
--Media Set Options  
    | MEDIANAME = { media_name | @media_name_variable }  
    | MEDIAPASSWORD = { mediapassword | @mediapassword_variable }  
    | BLOCKSIZE = { blocksize | @blocksize_variable }  
  
--Data Transfer Options  
    | BUFFERCOUNT = { buffercount | @buffercount_variable }  
    | MAXTRANSFERSIZE = { maxtransfersize | @maxtransfersize_variable }  
  
--Error Management Options  
    | { CHECKSUM | NO_CHECKSUM }  
    | { STOP_ON_ERROR | CONTINUE_AFTER_ERROR }  
  
--Monitoring Options  
    | STATS [ = percentage ]  
  
--Tape Options  
    | { REWIND | NOREWIND }  
    | { UNLOAD | NOUNLOAD }  
  
<replication_WITH_option> ::=  
    | KEEP_REPLICATION  
  
<change_data_capture_WITH_option> ::=  
    | KEEP_CDC  
  
<FILESTREAM_WITH_option> ::=  
    | FILESTREAM ( DIRECTORY_NAME = directory_name )
```

SQL 2014 - Módulo III

```
<service_broker_WITH_options> ::=  
| ENABLE_BROKER  
| ERROR_BROKER_CONVERSATIONS  
| NEW_BROKER  
  
<point_in_time_WITH_options-RESTORE_DATABASE> ::=  
| {  
    STOPAT = { 'datetime' | @datetime_var }  
| STOPATMARK = 'lsn:lsn_number'  
            [ AFTER 'datetime' ]  
| STOPBEFOREMARK = 'lsn:lsn_number'  
            [ AFTER 'datetime' ]  
}  
<point_in_time_WITH_options-RESTORE_LOG> ::=  
| {  
    STOPAT = { 'datetime' | @datetime_var }  
| STOPATMARK = { 'mark_name' | 'lsn:lsn_number' }  
            [ AFTER 'datetime' ]  
| STOPBEFOREMARK = { 'mark_name' | 'lsn:lsn_number' }  
            [ AFTER 'datetime' ] }
```

Vamos conhecer os tipos de verificações para restaurar um backup:

- **RESTORE HEADERONLY:** Obtém as informações do cabeçalho do backup. Caso o arquivo contenha mais de um backup, todas as informações serão retornadas. As informações retornadas são:
 - Nome e a descrição do arquivo;
 - O tipo de mídia que foi utilizada;
 - O tipo de backup escolhido no processo;
 - A data e hora em que o backup foi executado;
 - O tamanho do arquivo de backup;
 - O número sequencial do backup.

- **RESTORE FILELISTONLY:** Fornece informações a respeito dos arquivos de backup antes de uma eventual restauração desses arquivos. Fornece as seguintes informações:
 - O nome lógico e físico dos arquivos de dados e log;
 - O tamanho do conjunto de backup (**BACKUP SET**) em megabytes;
 - Informações dos grupos de arquivos (**FILEGROUPS**).
- **RESTORE LABELONLY:** Fornece informações a respeito das mídias de backup;
- **RESTORE VERIFYONLY:** Fornece informações a respeito da restauração dos backups existentes e se os arquivos estão corretos para o procedimento.

Vejamos os argumentos do comando **Restore**:

- **DATABASE:** Indica o tipo de restauração que se deseja realizar, neste caso, a restauração do banco de dados completa ou parcial;
- **{database_name}@var_database_name}**: Indica o nome do banco de dados a ser restaurado ou a variável que indica o nome do banco de dados (@**var_database_name**);
- **FROM:** Indica qual tipo de dispositivo será utilizado na restauração do backup. As opções **RECOVERY** ou **NORECOVERY** devem ser usadas em caso de não declararmos a opção **FROM**;
- **<backup_device>**: Indica o dispositivo de backup que deverá ser utilizado durante o processo de restauração. Trata-se da especificação do nome lógico que é feita com base na mesma regra definida para os dispositivos de backup criados a partir da system stored procedure **sp_addumpdevice**;
- **{DISK|TAPE}={‘physical_backup_device_name’}@physical_backup_device_name_var**: Tipo de dispositivo, disco (**DISK**) ou fita (**TAPE**), em que se encontram os arquivos de backup;

- **FILE=file_number**: Por meio deste número, podemos indicar qual conjunto de backup (**BACKUP SET**) deverá ser restaurado;
- **MEDIANAME={media_name}@var_media_name}**: Indica qual nome da mídia utilizada e esta deve ser idêntica ao nome da mídia do backup. Caso a especificação não seja atendida, a operação de restauração (**RESTORE**) apresentará erro e fracassará;
- **MOVE 'logical_file_name' TO 'operating_system_file_name'**: Esta opção permite que um backup seja restaurado em um novo arquivo no sistema operacional, porém, com o mesmo nome lógico. Isso se aplica quando restauramos um banco de dados feito em um backup em outro banco de dados com nome e/ou local diferente do original;
- **RECOVERY**: Esta opção indica que, ao final do processo de restauração (**RESTORE**), o banco de dados deve ser aberto e liberado para utilização. Indicado para restauração apenas de backup completo sem aplicação dos backups de log;
- **NORECOVERY**: Recomendado para restaurações (**RESTORE**) em que os backups de logs sejam restaurados após a restauração do backup completo/diferencial;
- **RESTART**: Opcão que permite que a restauração (**RESTORE**) do banco de dados seja reiniciada a partir do ponto em que foi interrompida;
- **RESTRICTED_USER**: Permite que apenas membros da system role **sysadmin**, **dbcreator** ou ainda **db_owner** do banco de dados possam acessá-lo após o processo de restauração (**RESTORE**);
- **STATS[=PERCENTAGE]**: Exibe uma mensagem assim que um percentual de Restore tenha sido atingido. O padrão dessa porcentagem é de 10% em caso de omissão deste valor;
- **<file_or_filegroup>**: Nome do arquivo ou do grupo de arquivos que será restaurado (**RESTORE**);

- **FILE={logical_file_name}@var_logical_file_name}**: Nome do arquivo lógico a ser restaurado;
- **FILEGROUP {logical_filegroup_name}@logical_file_group_name}**: Um ou mais grupos de arquivos que sofreram o procedimento de restauração (**RESTORE**);
- **LOG**: Restauração de backup de log, para recuperação completa ou recuperação em um ponto específico do tempo (**POINT IN-TIME RECOVERY**);
- **STOPAT**: Indica que a recuperação será feita em um ponto específico do tempo (**POINT IN-TIME RECOVERY**). Deve ser utilizada uma data e hora para essa opção.

5.11.2. Restauração de grupo de arquivos ou de arquivos de banco de dados

A restauração de arquivo ou de grupo de arquivos pode ser realizada em casos de backup completo ou de backup de arquivos ou de grupo de arquivos. Este tipo de recuperação pode ocorrer em função da perda física de um arquivo de dados ou em caso de necessidade de restauração de uma aplicação que esteja confinada em um arquivo ou grupo de arquivos específico.

A sintaxe para restauração de arquivos ou grupo de arquivos é a seguinte:

```
RESTORE DATABASE { database_name | @database_name_var }
    <file_or_filegroup> [ ,...n ]
    [ FROM <backup_device> [ ,...n ] ]
    WITH
    {
        [ RECOVERY | NORECOVERY ]
        [ , <general_WITH_options> [ ,...n ] ]
    } [ ,...n ]

    [ ; ]
```

A recuperação (**RESTORE**) neste caso será mais rápida do que se fosse a restauração completa.

5.11.3. Restauração de log de banco de dados

A restauração do log é importante para recuperação completa ou parcial do banco de dados. A sintaxe para restauração do arquivo log é a seguinte:

```
RESTORE LOG { database_name | @database_name_var }
[ <file_or_filegroup_or_pages> [ ,...n ] ]
[ FROM <backup_device> [ ,...n ] ]
[ WITH
{
    [ RECOVERY | NORECOVERY | STANDBY =
        {standby_file_name | @standby_file_name_var }
    ]
    | , <general_WITH_options> [ ,...n ]
    | , <replication_WITH_option>
    | , <point_in_time_WITH_options-RESTORE_LOG>
} [ ,...n ]
]
[ ; ]
```

Este tipo de recuperação se aplica apenas aos modelos de recuperação completa (**FULL**) e (**BULKED-LOGGED**).

5.11.4. Restauração de página de banco de dados

A restauração de página pode ser necessária em caso de corrompimento de páginas. Neste caso, em vez de restaurar o banco de dados inteiro ou dos arquivos de dados, podemos restaurar apenas as páginas danificadas.

A sintaxe para restauração das páginas é a seguinte:

```
RESTORE DATABASE { database_name | @database_name_var }
    PAGE = 'file:page [ ,...n ]'
    [ , <file_or_filegroups> ] [ ,...n ]
    [ FROM <backup_device> [ ,...n ] ]
    WITH
        NORECOVERY
    [ , <general_WITH_options> [ ,...n ] ][;]
```

Deve-se indicar o arquivo e as respectivas páginas que deverão ser restauradas. Este tipo de restauração somente é aplicável a modelos de recuperação completa (**FULL**) e (**BULKED-LOGGED**).

5.11.5. Restauração de bancos de dados de sistema (MSDB)

Os bancos de dados de sistema também podem precisar de recuperação e, para isso, precisam antes sofrer um processo de backup de caráter regular (exemplo diário). Eles podem, em alguns casos, ser reconstruídos ou restaurados. Isso vai depender do problema que esses bancos enfrentaram.

Em caso de perda e/ou falha, execute este comando:

```
DBCC CHECKDB
```

SQL 2014 - Módulo III

Caso o banco de dados tenha uma falha, ele será apontado como **SUSPECT**. Agora, se ocorrer a perda do banco de dados **MSDB**, verifique a versão do banco de dados **MSDB** que possui backup:

```
SELECT  
    SERVERPROPERTY('PRODUCTLEVEL') AS SP,  
    SERVERPROPERTY('PRODUCTVERSION') AS VERSION
```

O resultado será semelhante a este:

	SP	VERSION
1	SP1	11.0.3000.0

Caso o backup corresponda à versão, realize a restauração do backup do banco de dados desejado. Para isso, use o comando a seguir:

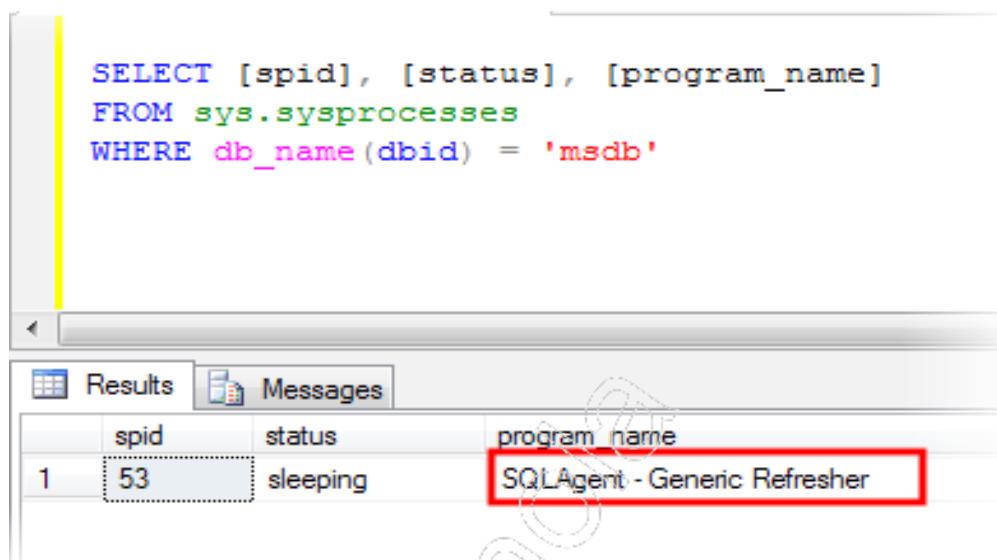
```
RESTORE HEADERONLY  
FROM DISK = N'arquivo de dados backup'  
GO
```

Certifique-se de que nenhum outro processo está utilizando o banco de dados **MSDB**. Caso as versões sejam iguais, realize a restauração do banco de dados **MSDB**.

```
SELECT spid, status, program_name  
FROM sys.sysprocesses  
WHERE db_name(dbid) = 'msdb'
```

Gerenciando a recuperação de dados

Observe o resultado a seguir:



	spid	status	program_name
1	53	sleeping	SQLAgent - Generic Refresher

Neste caso, devemos parar o service do SQL Server Agent utilizando o gerenciador de serviços do Windows:



Em seguida, podemos realizar o restore do backup do MSDB:

```
USE master
GO
RESTORE DATABASE [msdb]
FROM DISK = N'NOME DO ARQUIVO DE BACKUP'
WITH REPLACE
GO
```

Agora devemos reiniciar o serviço do SQL Server Agent utilizando o gerenciador de serviços do Windows:

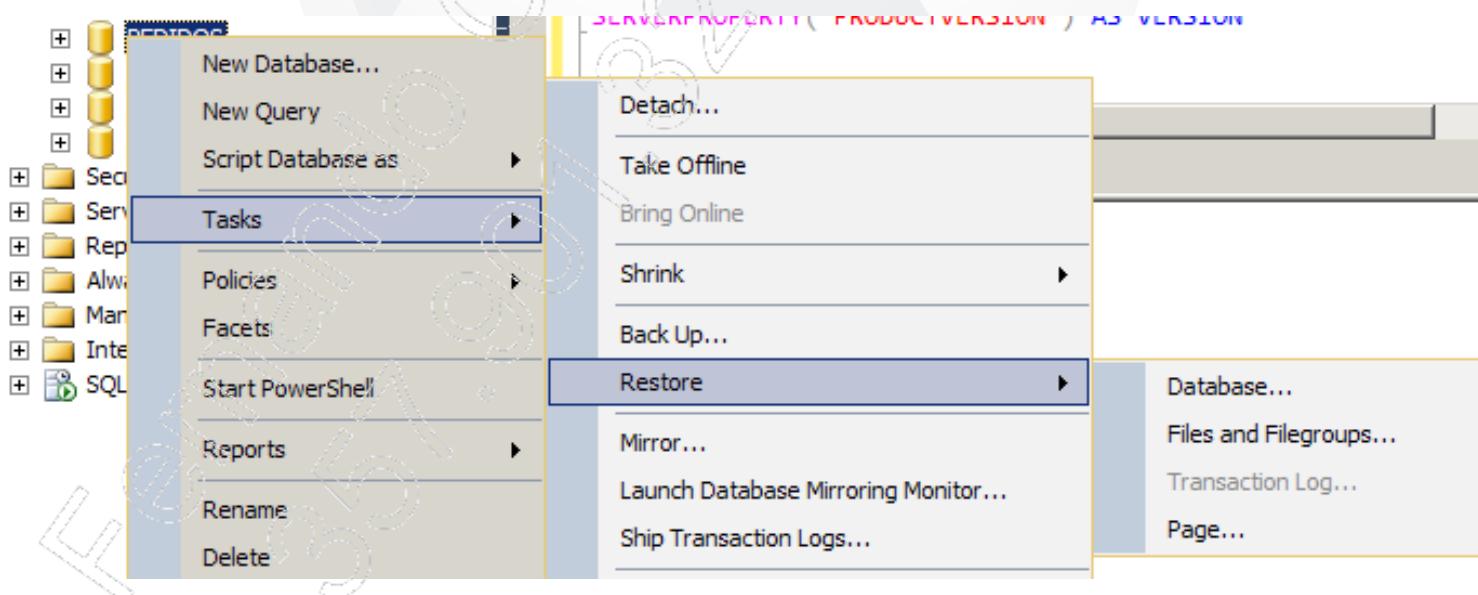


5.12. Restauração utilizando ambiente gráfico

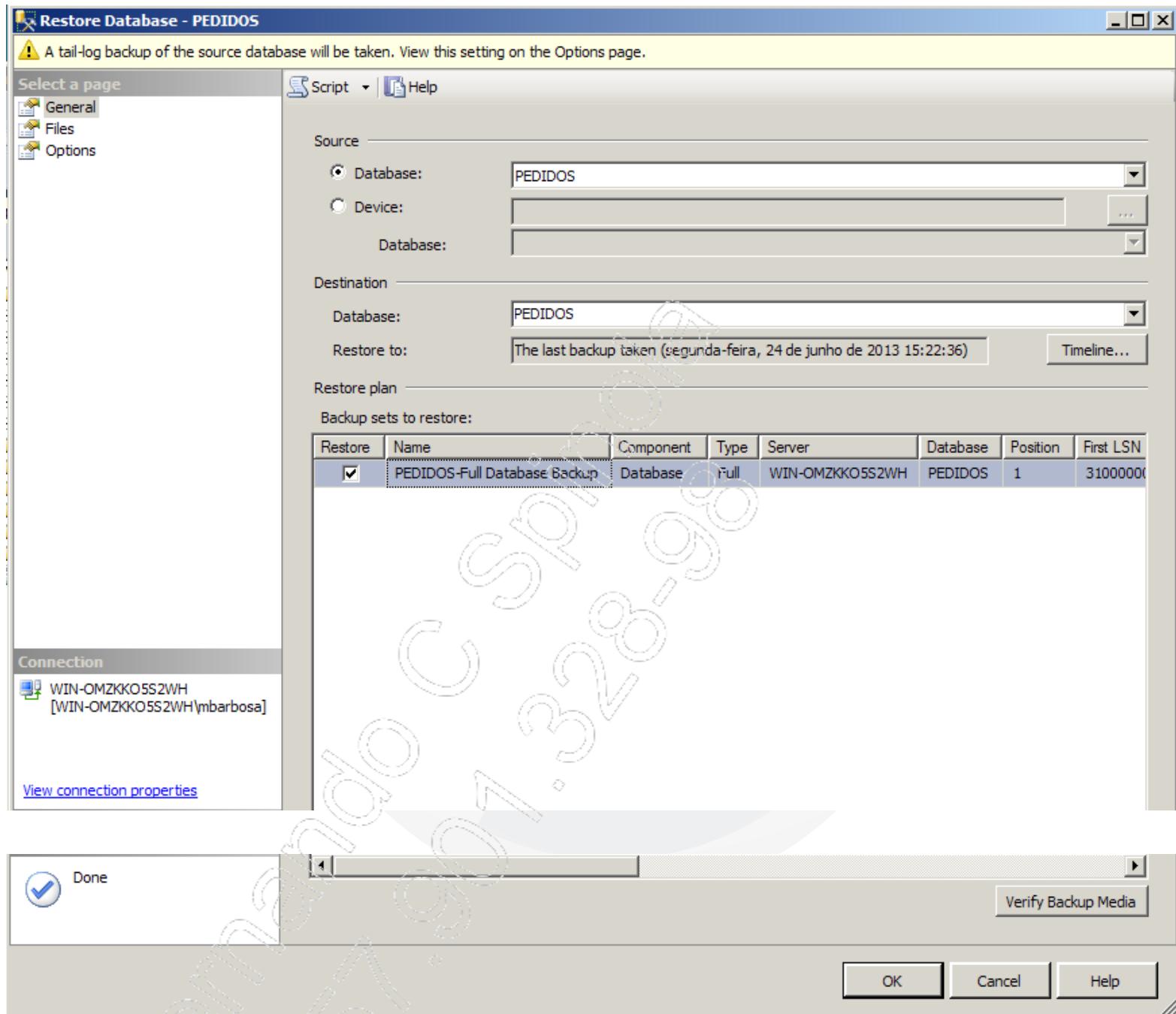
A restauração do banco de dados pode ser executada também através da interface gráfica do SSMS. Isso pode ser prático para execução pontual de restaurações. Ao final do procedimento, pode-se gerar um script SQL para execução do backup em modo texto.

Para executar a restauração em ambiente gráfico, realize os seguintes passos:

1. Posicione o mouse no banco de dados que deseja restaurar e selecione o menu **Restore**. Em seguida, o assistente de restauração abrirá a tela para seleção das opções de restauração;



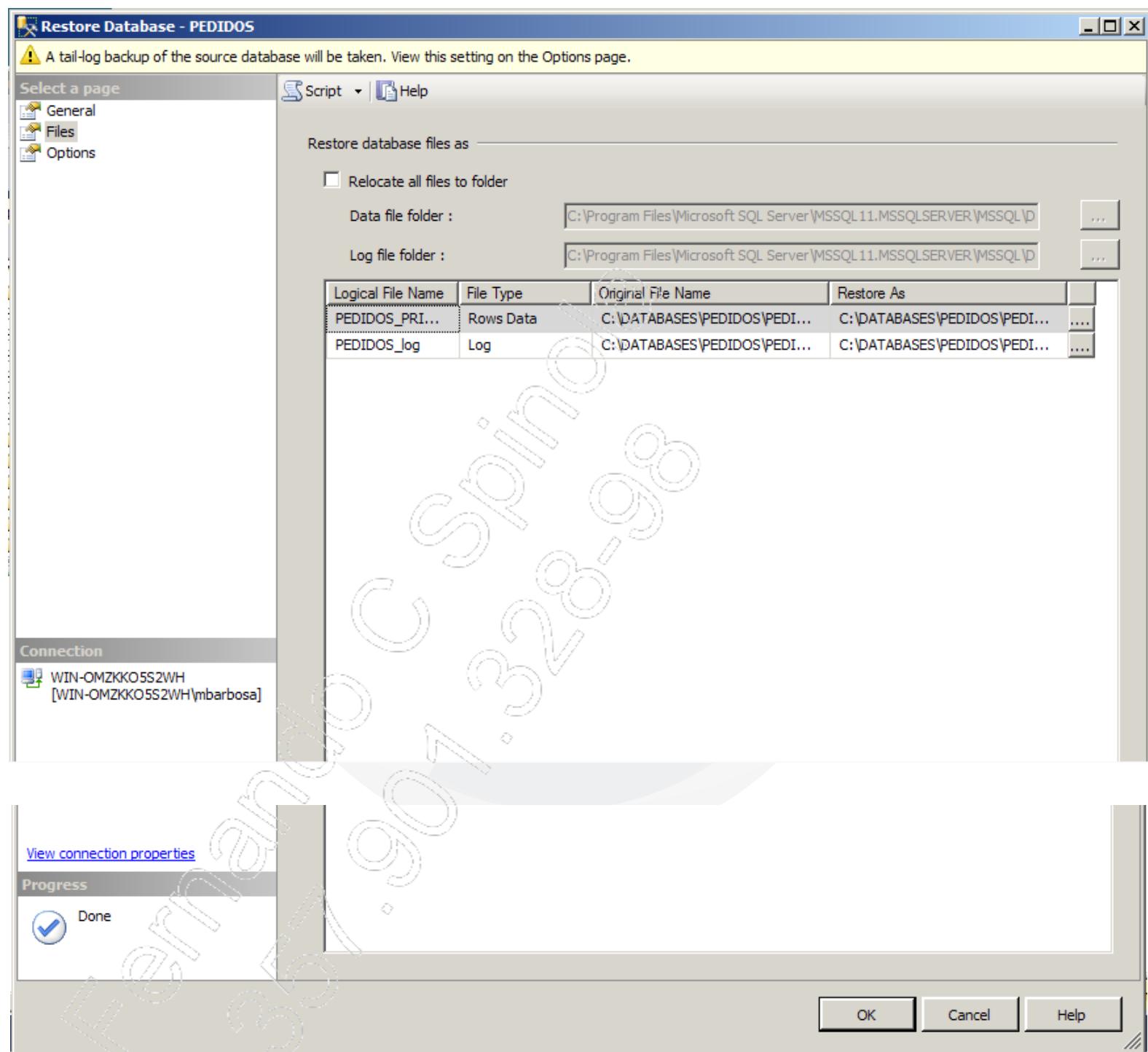
Gerenciando a recuperação de dados



2. Selecione o banco de dados desejado (**Database**);
3. Selecione o backup a ser restaurado (**Restore**);

SQL 2014 - Módulo III

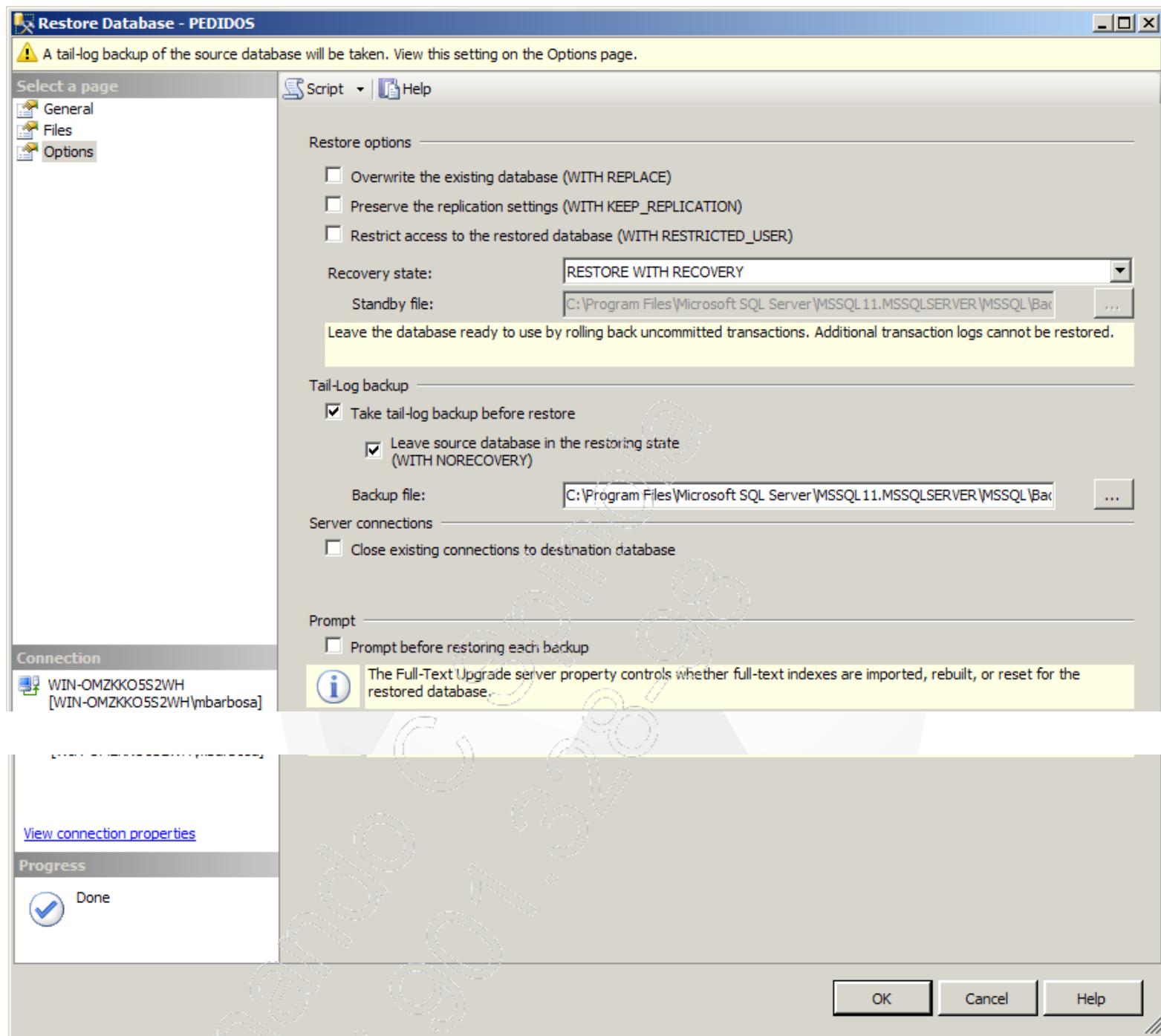
4. Selecione a opção **Files** para indicar os arquivos de dados que serão restaurados;



5. Caso haja necessidade de restaurar os arquivos em outro diretório, selecione a opção **Relocate all files to folder** e indique os diretórios para os arquivos de dados e de log;

Gerenciando a recuperação de dados

6. Selecione Options;



7. Selecione as opções **Overwrite the existing database**, para sobrepor o banco de dados caso ele exista;

8. Selecione a opção de recovery state (**restore with norecovery**) para restaurar um backup e depois os backups de log que ele tiver para restaurações completas e incompletas (recuperação no ponto específico no tempo). Use a opção **Restore with recovery** para restaurar o banco de dados e realizar a abertura, desconsiderando os backups de logs realizados;

9. Selecione a opção de tail-log backup, caso efetue a restauração no mesmo banco de dados que realizou backup e utilize a opção (**restore with no recovery**).

5.13. Anexando e desanexando um banco de dados

Todos os bancos de dados podem ser anexados e desanexados de uma instância. Isso pode ocorrer por vários motivos, entre eles:

- Movimentação do banco de dados para um novo servidor;
- Reorganização de instância;
- Backup frio de um banco de dados.

Para desanexar um banco de dados existente em uma instância, é necessário encerrar o banco de dados e parar todos os processos que estão sendo executados nesse banco.

Para anexar um arquivo de dados, devemos executar a system stored procedure **sp_attach_db** e, para desanexar, a **sp_detach_db**:

```
sp_attach_db @dbname = N'nome database',
              @filename1 = N'nome arquivo de dados',
              @filename2 = N'nome arquivo de log'

sp_detach_db @dbname = N'nome database'
```

Desanexando um banco de dados, ele deixará de pertencer à instância.

Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo.

- Todo procedimento de backup deve ser embasado por uma política de backups que deve contemplar o que se deseja proteger, quais cenários ela abrange e a política de testes de restauração de backup;
- Existem dois tipos de backup: o físico e o lógico. O lógico contempla apenas a cópia de dados de forma inconsistente. Neste modelo, podemos copiar tabelas individualmente. O tipo de backup físico contempla a cópia física (arquivos de dados e log) do banco de dados;
- O backup físico pode ser quente (**Hot Backup**) ou frio (**Cold Backup**). O backup quente é feito sem a necessidade de encerramento do banco de dados. Este tipo de backup tem como característica a inconsistência, pois, entre o início e o fim do backup, podem ter ocorrido alterações na base de dados. Essas alterações somente serão aplicadas caso seja restaurado o backup de log subsequente ao backup completo. Já o backup frio tem como característica a consistência, pois não há necessidade de recuperação dos backups de log;
- O modo de recuperação **FULL** é o mais recomendado para aplicações transacionais (OLTP), pois permitem a recuperação completa e em um ponto específico do tempo (**Point in-time recovery**);
- O modo de recuperação **BULKED-LOGGED** é recomendado para aplicações que utilizam os comandos **BCP** e **BULK INSERT** e não é recomendado para sistemas transacionais;
- O modo de recuperação **SIMPLE** é utilizado pelos bancos de dados de sistema do SQL Server e deve ser adotado apenas por sistemas que tolerem a perda de dados;

- O mecanismo para liberação de espaço ocupado pelo log pode ser feito de duas formas: por meio dos mecanismos de backup de log e compactação do arquivo, ou através da mudança do modo de recuperação para modo **SIMPLE**, compactação do arquivo de log e volta ao modo de recuperação **FULL**;
- A estratégia de recuperação é fundamental para a rotina de backup e deve ser testada periodicamente pelo administrador do banco de dados;
- É importante catalogar e ter as informações de backup sempre disponíveis;
- Cuidado com os scripts de backup e de restore, pois são fundamentais para a política de backup, assim como os procedimentos propriamente ditos.

5

Gerenciando a recuperação de dados

Teste seus conhecimentos

Fernando
357.901-3239



IMPACTA
EDITORA

1. Por que o modelo de recuperação simples não é recomendado para sistemas transacionais?

- a) Porque é usado nos bancos de dados de sistema.
- b) Somente bancos de dados de sistema podem utilizar esse modelo de recuperação.
- c) O enunciado da questão não procede, pois o modelo de recuperação simples deve ser usado em sistemas transacionais.
- d) Porque não podemos retornar um ponto específico de dados, somente o último backup FULL.
- e) Nenhuma das alternativas anteriores está correta.

2. Qual afirmação está correta sobre backup frio e backup quente?

- a) Backup frio é utilizado quando o banco está em uso.
- b) Não há diferença entre esses tipos de backup.
- c) Somente podemos utilizar o backup quente.
- d) Backup quente somente é executado em um modelo de recuperação FULL.
- e) No backup frio o banco fica “indisponível”, ao contrário do quente.

3. O que é um device de backup do SQL Server?

- a) É o local de armazenamento em disco.
- b) É um tipo de backup.
- c) É a forma de armazenamento do backup FULL.
- d) É a forma de armazenamento do backup de LOG.
- e) É o local de armazenamento de backup.

4. Ao planejarmos um backup, o que não é correto afirmar?

- a) O backup pode ser realizado junto com as atividades.
- b) É necessário prever o volume dos bancos de dados.
- c) O volume das bases não interfere no backup.
- d) Planeje o tempo de retenção das mídias.
- e) Defina quais usuários realizarão o backup.

5. Quais usuários podem realizar o backup?

- a) Como o backup é importante, qualquer usuário.
- b) Somente usuários da role SYSADMIN.
- c) Usuários que pertençam às roles SYSADMIN e DB_OWNER.
- d) Usuários que pertençam às roles DB_OWNER e DB_BACKUPOPERATOR.
- e) Usuários das roles DB_BACKUPOPERATOR, DB_OWNER e SYSADMIN.

Gerenciando a recuperação de dados

Mãos à obra!

5

Fernando Spahno
357.907.28-0000



IMPACTA
EDITORA

Laboratório 1

Observações iniciais

1 – O que será feito neste laboratório

Neste laboratório, vamos executar uma sequência de backups no database chamado **Colegio**. Para tanto, teremos que realizar este exercício em etapas:

- Ativar o uso da procedure **xp_cmdshell**;
- Criar diretórios para o database e para o device de backup;
- Criar o database **Colegio**;
- Criar um device de backup para armazenar os backups desse database;
- Implementar uma estratégia de backups;
- Verificar os backups dentro do device de backup;
- Simular uma falha no database;
- Restaurar o database **Colegio** de acordo com os backups feitos anteriormente.

2 – Scripts utilizados neste laboratório

A pasta **Capítulo_05** contém o **Script_01** com os comandos necessários para a realização deste laboratório. Utilize os scripts se não desejar digitar os respectivos comandos ou apenas para corrigir sua execução.

A – Criando diretórios para o database e para o device de backup

1. Clique no botão **Start** selecione **All Programs**, escolha **Microsoft SQL Server 2014** e, em seguida, clique em **SQL Server Management Studio**;
2. Conecte-se ao **SQL Server** com a autenticação do Windows;
3. Do lado esquerdo da tela, no **Object Explorer**, expanda a pasta **Databases**;
4. Na barra de ferramentas, clique na opção **New Query**;
5. Escreva o comando adiante:

```
Exec xp_cmdshell 'MD C:\Bancos\colegio'  
go  
Exec xp_cmdshell 'MD C:\Backup\colegio'  
Go
```

A primeira linha cria, na raiz do disco **C:**, uma pasta chamada **Bancos\colegio**, que será o diretório no qual ficarão os arquivos de dados e de log do database que será criado mais adiante.

A segunda linha cria, também na raiz do disco **C:**, uma pasta chamada **Backup\colegio**, que é o diretório no qual ficará armazenado o device de backup deste database.

B – Criando o database Colegio

1. Execute o comando a seguir para criar o database **Colegio**:

```
CREATE DATABASE Colegio
ON PRIMARY
(
    Name      = 'Colegio_primary',
    filename   = 'C:\Bancos\colegio\Colegio_primary.mdf',
    size       = 10MB,
    maxsize    = 100MB,
    filegrowth = 10MB
)
LOG ON
(
    Name      = 'Colegio_Log',
    filename   = 'C:\Bancos\colegio\Colegio_Log.ldf',
    size       = 10MB,
    maxsize    = 10MB,
    filegrowth = 10MB
)

-- Altere o modo de Recovery para FULL

Alter database colegio set recovery full
```

C – Criando um device de backup para armazenar os backups deste database

1. Execute o comando a seguir para criar o device de backup:

```
Exec sp_addumpdevice
@devtype      = 'Disk',
@logicalname = 'Backupcolegio',
@physicalname = 'C:\Backup\colegio\colegio.bak'
```

D – Implementando uma estratégia de backups

1. Acesse o database **Colegio** e crie dentro dele as tabelas **Materia**, **Professor**, **Aluno** e **Funcionario**, como mostram os comandos a seguir:

```
Use Colegio
go
CREATE TABLE Materia
(
Cod_Mat int,
Nome_Mat char(30)
)
go
CREATE TABLE Professor
(
Cod_Prof int,
Nome_Prof char(30)
)
go
CREATE TABLE Aluno
(
Cod_Aluno int,
Nome_Aluno char(30)
)
go
CREATE TABLE Funcionario
(
Cod_Func int,
Nome_Func char(30)
)
go
```

SQL 2014 - Módulo III

2. Escreva os comandos a seguir para inserir dados nas tabelas do database **Colegio**:

```
INSERT Materia VALUES(1,'Matemática')
INSERT Materia VALUES(2,'Física')
go
INSERT Professor VALUES(1,'Leandro')
INSERT Professor VALUES(2,'Marcos')
go
INSERT Aluno VALUES(1,'André')
INSERT Aluno VALUES(2,'José')
go
INSERT Funcionario VALUES(1,'Paulo')
INSERT Funcionario VALUES(2,'Paula')
go
Checkpoint
Go
```

3. Escreva o comando a seguir para executar o primeiro **Backup Full** do database **Colegio**:

```
BACKUP DATABASE Colegio
TO Backupcolegio
WITH
DESCRIPTION = 'Primeiro Backup Full do Database Colegio',
INIT,
MEDIANAME = 'Backup Colegio',
NAME = 'Backup Full Colegio'
```

4. Insira mais dados nas tabelas do database **Colegio**, como mostram os comandos seguintes:

```
INSERT Materia VALUES(3,'Química')
INSERT Materia VALUES(4,'Astronomia')
go
INSERT Professor VALUES(3,'Paulo')
INSERT Professor VALUES(4,'Roberta')
go
INSERT Aluno VALUES(3,'Renato')
INSERT Aluno VALUES(4,'Ricardo')
go
INSERT Funcionario VALUES(3,'Mauro')
INSERT Funcionario VALUES(4,'Maura')
go
Checkpoint
go
```

5. Execute o comando adiante para fazer o backup do **Transaction Log** do database:

```
BACKUP LOG Colegio
TO Backupcolegio
WITH
DESCRIPTION = 'Primeiro Backup Log do Database Colegio',
NOINIT,
MEDIANAME = 'Backup Colegio',
NAME = 'Backup Log 1 Colegio'
```

SQL 2014 - Módulo III

6. Insira os dados nas tabelas, conforme os comandos a seguir:

```
INSERT Materia VALUES(5,'Astrologia')
INSERT Materia VALUES(6,'Quiromancia')
go
INSERT Professor VALUES(5,'Morgana')
INSERT Professor VALUES(6,'Odavlas')
go
INSERT Aluno VALUES(5,'Agnes')
INSERT Aluno VALUES(6,'Agnus')
go
INSERT Funcionario VALUES(5,'Marcelo')
INSERT Funcionario VALUES(6,'Mauricio')
go
Checkpoint
Go
```

7. Execute o comando adiante, que realiza mais um backup do **Transaction Log** do database:

```
BACKUP LOG Colegio
TO Backupcolegio
WITH
DESCRIPTION = 'Segundo Backup Log do Database Colegio',
NOINIT,
MEDIANAME = 'Backup Colegio',
NAME = 'Backup Log 2 Colegio'
```

8. Insira mais dados nas tabelas, como indicam os comandos a seguir:

```
INSERT Materia VALUES(7,'Tarô')
INSERT Materia VALUES(8,'Cartomancia')
go
INSERT Professor VALUES(7,'Zoroastra')
INSERT Professor VALUES(8,'Zoroastro')
go
INSERT Aluno VALUES(7,'Eleotério')
INSERT Aluno VALUES(8,'Midas')
go
INSERT Funcionario VALUES(7,'Cristina')
INSERT Funcionario VALUES(8,'Cristiano')
go
Checkpoint
```

9. Execute o comando a seguir para realizar um backup diferencial do database **Colegio**:

```
BACKUP DATABASE Colegio
TO Backupcolegio
WITH
DIFFERENTIAL,
DESCRIPTION = 'Backup Diferencial do database Colegio',
NOINIT,
MEDIANAME = 'Backup Colegio',
NAME = 'Backup Diferencial Colegio'
```

10. Insira os dados a seguir nas tabelas do database:

```
INSERT Materia VALUES(9,'Telepatia')
INSERT Materia VALUES(10,'Runas')
go
INSERT Professor VALUES(9,'Rubeo')
INSERT Professor VALUES(10,'Nana Sara')
go
INSERT Aluno VALUES(9,'Silvia')
INSERT Aluno VALUES(10,'Sonia')
go
INSERT Funcionario VALUES(9,'Bianca')
INSERT Funcionario VALUES(10,'Beatriz')
go
Checkpoint
```

11. Execute o comando a seguir para realizar o último backup do **Transaction Log** do database:

```
BACKUP LOG Colegio
TO Backupcolegio
WITH
DESCRIPTION = 'Terceiro Backup Log do Database Colegio',
NOINIT,
MEDIANAME = 'Backup Colegio',
NAME = 'Backup Log 3 Colegio'
```

E – Verificando os backups dentro do device de backup

1. Execute o comando a seguir para saber quais são os backups que foram criados dentro do device:

```
RESTORE HEADERONLY  
FROM DISK = 'C:\Backup\colegio\colegio.bak'
```

F – Simulando uma falha no database

1. Execute os comandos a seguir para simular um erro, gerando a necessidade de restaurar dados de uma estratégia de backup:

```
Use Master  
go  
DROP DATABASE Colegio  
go  
SELECT * FROM sys.databases
```

G – Restaurando o database Colegio de acordo com os backups feitos anteriormente

1. Execute os comandos a seguir para restaurar os dados dos respectivos backups e recuperar o database **Colegio**:

```
RESTORE DATABASE Colegio
FROM Backupcolegio
WITH FILE = 1, NORECOVERY
go
RESTORE DATABASE Colegio
FROM Backupcolegio
WITH FILE = 4, NORECOVERY
go
RESTORE LOG Colegio
FROM Backupcolegio
WITH FILE = 5, RECOVERY
go
```

2. Acesse novamente o database **Colegio** e verifique, executando os comandos adiante, que o database foi recuperado como se desejava:

```
Use Colegio
go
SELECT * FROM Materia
SELECT * FROM Aluno
SELECT * FROM Professor
SELECT * FROM Funcionario
```

3. Para terminar, feche o **Microsoft SQL Server Management Studio**.

Laboratório 2

Observações iniciais

1 – O que será feito neste exercício

Neste laboratório, vamos executar uma sequência de backups no database chamado **Academia**. É fundamental destacar que essa sequência de backups será feita graficamente.

Para tanto, vamos ter que executar o exercício em algumas etapas:

- Criar diretórios para o database e para o device de backup;
- Criar o database **Academia**;
- Criar um device de backup para armazenar os backups deste database;
- Implementar uma estratégia de backup;
- Verificar os backups dentro do device de backup;
- Simular uma falha no database;
- Restaurar o database **Academia** de acordo com os backups feitos anteriormente.

2 – Scripts utilizados neste laboratório

A pasta **Capítulo_05** contém o **Script_02** com os comandos necessários para a realização deste laboratório. Utilize os scripts se não desejar digitar os respectivos comandos ou apenas para corrigir sua execução.

A – Executando uma sequência de backups feita graficamente no database Academia

1. Clique no botão **Start**, em **All Programs** e, em seguida, em **Microsoft SQL Server 2014** e escolha **Microsoft SQL Server Management Studio**;
2. Conecte-se no **SQL Server** com a autenticação do Windows;
3. Expanda a pasta **Databases**;
4. Na barra de ferramentas, clique na opção **New Query**;
5. Escreva e execute os seguintes comandos para criar os diretórios em que ficarão os arquivos do database **Academia** e o arquivo de **Backup**:

```
Exec xp_cmdshell 'MD C:\Bancos\Academia'  
go  
Exec xp_cmdshell 'MD C:\Backup\Academia'  
go
```

SQL 2014 - Módulo III

6. Ainda utilizando o **Microsoft SQL Server Management Studio**, crie o database **Academia**, como mostra o comando a seguir:

```
CREATE DATABASE Academia
ON PRIMARY
(
    name      = 'Academia_Dados',
    filename  = 'C:\Bancos\Academia\Academia_Dados.mdf',
    size      = 10MB,
    maxsize   = 100MB,
    filegrowth = 10MB
)
LOG ON
(
    name      = 'Academia_Log',
    filename  = 'C:\Bancos\Academia\Academia_Log.ldf',
    size      = 10MB,
    maxsize   = 10MB,
    filegrowth = 10MB
)
GO
Alter database academia set recovery FULL
GO
```

7. Acesse o database **Academia** e crie nele as tabelas, como mostram os comandos a seguir:

```
Use Academia
go
CREATE TABLE Materia
(
Cod_Mat int,
Nome_Mat char(30)
)
go
CREATE TABLE Professor
(
Cod_Prof int,
Nome_Prof char(30)
)
go
CREATE TABLE Aluno
(
Cod_Aluno int,
Nome_Aluno char(30)
)
go
CREATE TABLE Funcionario
(
Cod_Func int,
Nome_Func char(30)
)
go
```

SQL 2014 - Módulo III

8. No database **Academia**, insira os dados nas tabelas, como mostram os comandos a seguir:

```
--Primeira Parte

INSERT Materia VALUES(1,'Matemática')
INSERT Materia VALUES(2,'Física')
go
INSERT Professor VALUES(1,'Leandro')
INSERT Professor VALUES(2,'Marcos')
go
INSERT Aluno VALUES(1,'André')
INSERT Aluno VALUES(2,'José')
go
INSERT Funcionario VALUES(1,'Paulo')
INSERT Funcionario VALUES(2,'Paula')
go
Checkpoint
go
```

9. Crie o **device** em que o backup ficará armazenado. Para isso, ainda utilizando a ferramenta **Microsoft SQL Server Management Studio**, do lado esquerdo da tela, na área **Object Explorer**, expanda a pasta **Server Objects**;

10. Clique com o botão direito sobre a pasta **Backup Device** e escolha a opção **New Backup Device**;

11. Na tela que será exibida, no campo **Device Name**, escreva **BackupAcademia**;

12. Com a opção **File** checada, clique nas reticências (...) e, na tela que será exibida, escolha o diretório **C:\Backup\Academia\BackupAcademia.bak**;

13. Clique no botão **OK**;

14. Observe, no **Object Explorer**, que o device de backup foi criado;

15. Expanda a pasta **Databases** e, se o database **Academia** não estiver na lista de bancos, clique com o botão direito do mouse sobre a pasta **Databases** e escolha a opção **Refresh**;
16. Clique com o botão direito do mouse sobre o database **Academia** e escolha a opção **Tasks**. Em seguida, escolha a opção **Back Up**;
17. Observe que, no campo **Database**, já está selecionado o database **Academia** e que, no campo **Backup Type**, está selecionada a opção **Full**. Note também que, na área **Backup Component**, está selecionada a opção **Database**. Na área **Backup Set**, no campo **Name**, está escrito **Academia-Full Database Backup**;
18. No campo **Description** desta tela, escreva **Primeiro Backup Full**;
19. Observe que, no campo **Destination**, já vem escrito o endereço **Default** para a localização deste **Backup**. Clique no botão **Remove**, do lado direito da tela, e, em seguida, clique no botão **Add**;
20. Na tela que será exibida, clique na opção **Backup Device**;
21. Já deverá estar selecionado o nome do device **BackupAcademia**. Se não estiver, clique na seta ao lado e selecione o device **BackupAcademia**;
22. Clique em **OK**;
23. Note que, no campo **Destination**, ficou registrado o device **BackupAcademia**;
24. Clique no botão **OK**;
25. Após alguns segundos, deverá aparecer uma mensagem dizendo que o backup foi feito com sucesso. Clique no botão **OK**;

SQL 2014 - Módulo III

26. Escreva e execute os comandos a seguir para inserir mais dados nas tabelas desse database:

```
Use Academia
go

--Segunda Parte

INSERT Materia VALUES(3,'Química')
INSERT Materia VALUES(4,'Astronomia')
go
INSERT Professor VALUES(3,'Paulo')
INSERT Professor VALUES(4,'Roberta')
go
INSERT Aluno VALUES(3,'Renato')
INSERT Aluno VALUES(4,'Ricardo')
go
INSERT Funcionario VALUES(3,'Mauro')
INSERT Funcionario VALUES(4,'Maura')
go
Checkpoint
go
```

27. Do lado esquerdo da tela, no **Object Explorer**, clique com o botão direito do mouse sobre o database **Academia**, escolha a opção **Tasks** e, em seguida, **Back Up**;

28. Na tela que será exibida, no campo **Backup Type**, deve estar escrita a opção **Full**. Escolha a opção **Transaction Log**;

29. No campo **Description**, escreva **Primeiro Backup Log**;

30. Observe que, no campo **Destination**, está escrito **BackupAcademia**;

31. Clique no botão **OK**;

32. Após alguns segundos, deverá aparecer uma mensagem dizendo que o backup foi feito com sucesso. Clique no botão **OK**;

Gerenciando a recuperação de dados

33. Insira novamente os dados nas tabelas desse database. Para tanto, escreva e execute os comandos a seguir:

```
-- Terceira Parte

Use Academia
go
INSERT Materia VALUES(5,'Astrologia')
INSERT Materia VALUES(6,'Quiromancia')
go
INSERT Professor VALUES(5,'Morgana')
INSERT Professor VALUES(6,'Odavlas')
go
INSERT Aluno VALUES(5,'Agnes')
INSERT Aluno VALUES(6,'Agnus')
go
INSERT Funcionario VALUES(5,'Marcelo')
INSERT Funcionario VALUES(6,'Mauricio')
go
Checkpoint
go
```

34. Do lado esquerdo da tela, no **Object Explorer**, clique com o botão direito do mouse sobre o database **Academia**, escolha a opção **Tasks** e, em seguida, **Back Up**;

35. Na tela que será exibida, no campo **Backup Type**, deve estar escrita a opção **Full**. Escolha a opção **Transaction Log**;

36. No campo **Description**, escreva **Segundo Backup Log**;

37. Observe que, no campo **Destination**, está escrito **BackupAcademia**;

38. Clique no botão **OK**;

39. Após alguns segundos, deverá aparecer uma mensagem dizendo que o backup foi feito com sucesso. Clique no botão **OK**;

SQL 2014 - Módulo III

40. Insira novamente os dados nas tabelas desse database. Para tanto, escreva e execute os comandos a seguir:

```
-- Quarta Parte

Use Academia
go
INSERT Materia VALUES(7,'Taro')
INSERT Materia VALUES(8,'Cartomancia')
go
INSERT Professor VALUES(7,'Zoroastra')
INSERT Professor VALUES(8,'Zoroastro')
go
INSERT Aluno VALUES(7,'Eleotério')
INSERT Aluno VALUES(8,'Midas')
go
INSERT Funcionario VALUES(7,'Cristina')
INSERT Funcionario VALUES(8,'Cristiano')
go
Checkpoint
go
```

41. Do lado esquerdo da tela, no **Object Explorer**, clique com o botão direito do mouse sobre o database **Academia**, escolha a opção **Tasks** e, em seguida, **Back Up**;

42. Na tela que será exibida, no campo **Backup Type**, deve estar escrita a opção **Full**. Escolha a opção **Differential**;

43. No campo **Description**, escreva **Backup Diferencial**;

44. Observe que, no campo **Destination**, está escrito **BackupAcademia**;

45. Clique no botão **OK**;

46. Após alguns segundos, deverá aparecer uma mensagem dizendo que o backup foi feito com sucesso. Clique no botão **OK**;

Gerenciando a recuperação de dados

47. Insira novamente os dados nas tabelas desse database. Para isso, escreva e execute os comandos a seguir:

```
-- Quinta Parte  
Use Academia  
go  
INSERT Materia VALUES(9,'Telepatia')  
INSERT Materia VALUES(10,'Runas')  
go  
INSERT Professor VALUES(9,'Rubeo')  
INSERT Professor VALUES(10,'Nana Sara')  
go  
INSERT Aluno VALUES(9,'Silvia')  
INSERT Aluno VALUES(10,'Sonia')  
go  
INSERT Funcionario VALUES(9,'Bianca')  
INSERT Funcionario VALUES(10,'Beatriz')  
go  
Checkpoint  
go
```

48. Do lado esquerdo da tela, no **Object Explorer**, clique com o botão direito do mouse sobre o database **Academia**, escolha a opção **Tasks** e, em seguida, **Back Up**;

49. Na tela que será exibida, no campo **Backup Type** deve estar escrita a opção **Full**. Escolha a opção **Transaction Log**;

50. No campo **Description**, escreva **Terceiro Backup Log**;

51. Observe que, no campo **Destination**, está escrito **BackupAcademia**;

52. Clique no botão **OK**;

53. Após alguns segundos, deverá aparecer uma mensagem dizendo que o backup foi feito com sucesso. Clique no botão **OK**;

SQL 2014 - Módulo III

54. Execute o comando a seguir para verificar quantos registros cada tabela possui:

```
Use Academia  
go  
SELECT * FROM Materia  
SELECT * FROM Professor  
SELECT * FROM Aluno  
SELECT * FROM Funcionario
```

55. Para simular um problema com o database **Academia**, escreva e execute os comandos a seguir:

```
Use Master  
go  
DROP DATABASE Academia
```

56. Para restaurar graficamente o database, clique com o botão direito do mouse sobre o database **Academia**, escolha a opção **Tasks** e, em seguida, selecione **Restore e Database**;

57. Na tela que será exibida, no campo **Source:Database**, escreva **Academia**;

58. Note que, na opção **Destination: Database**, deverá estar escrito **Academia**.

59. Note também que, na área **Backup sets to restore**, deverá aparecer três registros de backup:

- O Backup Full;
- O Backup do Transaction Log;
- O backup diferencial.

60. Note também que esses três registros deverão estar marcados;
61. Clique no botão **OK**;
62. Após alguns segundos, deve aparecer uma mensagem dizendo que o processo de **restore** ocorreu com sucesso. Clique no botão **OK**;
63. Para encerrar, escreva e execute os comandos a seguir, verificando se todos os dados foram restaurados com sucesso:

```
Use Academia
go
SELECT * FROM Materia
SELECT * FROM Aluno
SELECT * FROM Professor
SELECT * FROM Funcionario
go
```

64. Feche o **Microsoft SQL Server Management Studio**.

Laboratório 3

Observações iniciais

1 - O que será feito neste exercício

Neste laboratório, vamos executar um **Restore** de um **Backup** do database **Pubs** que foi feito pela versão 2008 do Microsoft SQL Server.

2 - Scripts utilizados neste laboratório

A pasta **Capítulo_05** contém o **Script_03** com os comandos necessários para a realização deste laboratório. Utilize os scripts se não desejar digitar os respectivos comandos ou apenas para corrigir sua execução.

A - Executando restore de um backup do database Pubs

1. Abra o **Microsoft SQL Server Management Studio**;
2. Conecte-se ao SQL Server com a autenticação do Windows;
3. Para criar o device de backup acessando o **BackupPubs** localizado na pasta **C:\Backup**, escreva e execute o comando a seguir:

```
Exec sp_addumpdevice
@devtype      = 'Disk',
@logicalname = 'BackupPubs',
@physicalname = 'C:\Backup\BackupPubs.BAK'
go
```

4. Crie um database chamado **Pubs2008**, executando o comando a seguir:

```
CREATE DATABASE Pubs2008  
go
```

5. Para restaurar o backup do database **Pubs** feito com o SQL Server 2008, execute o comando a seguir:

```
RESTORE DATABASE Pubs2008  
FROM BackupPubs  
WITH REPLACE
```

6. Assim que o backup terminar a sua execução, execute o comando a seguir para ler os dados de uma das tabelas do database **Pubs2008**:

```
Use Pubs2008  
go  
SELECT * FROM authors  
go
```

7. Do lado esquerdo da tela, no **Object Explorer**, clique com o botão direito do mouse sobre a pasta **Databases** e escolha a opção **Refresh**;

8. Expanda o database **Pubs2008** e a pasta **Tables**;

9. Observe que as tabelas do database **Pubs** agora fazem parte do database **Pubs2008**, assim como suas **views**, **procedures** e **user defined datatypes**;

10. Para encerrar, feche o **Microsoft SQL Server Management Studio**.

Transferência e manipulação de dados

6

- ✓ Exportando e importando dados;
- ✓ Ferramenta de transformação e cópia de dados.

Fernando Spinola
357.907.3989



IMPACTA
EDITORA

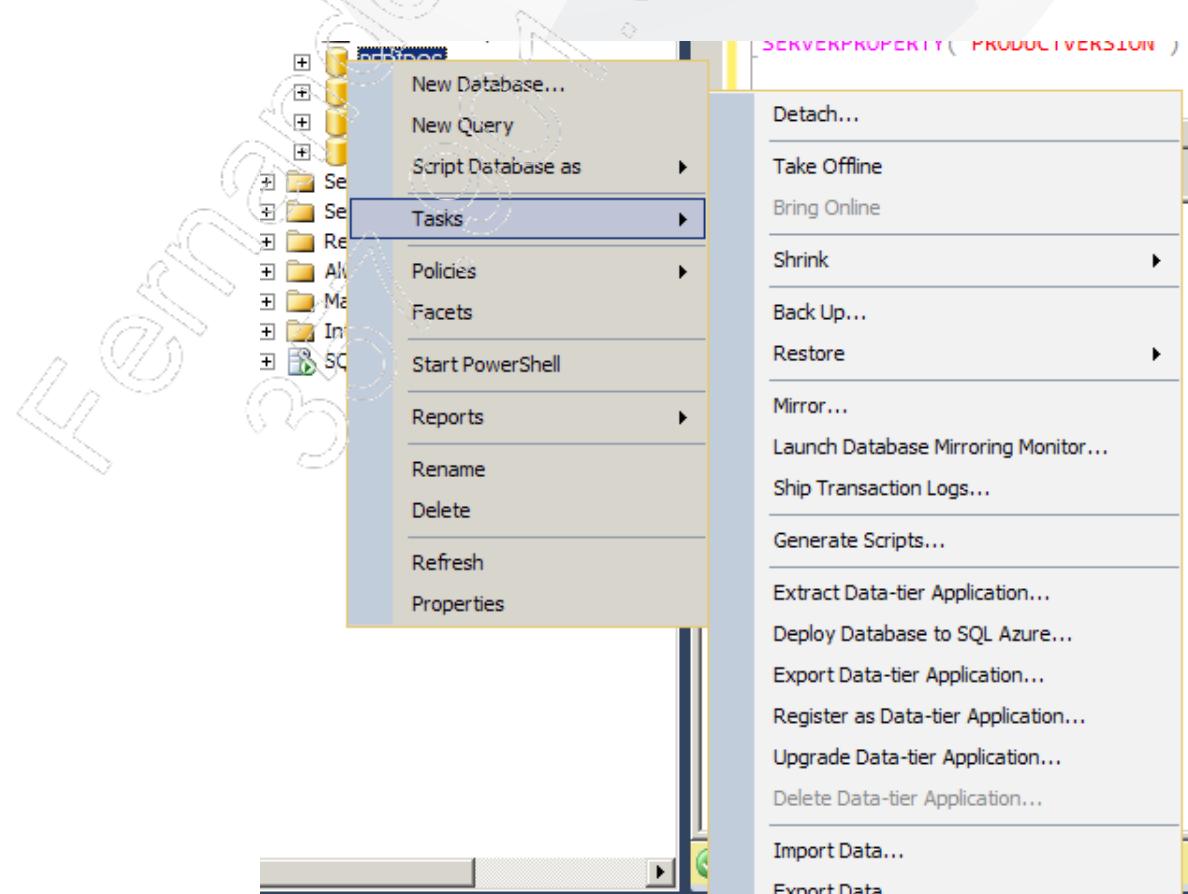
6.1. Introdução

Já vimos o que é o gerenciamento de backup e tratamos do backup físico. Neste capítulo, abordaremos a manipulação e a exportação de dados, também conhecidas como backup lógico. Veremos, inclusive, as formas e ferramentas para exportação e cópia de dados.

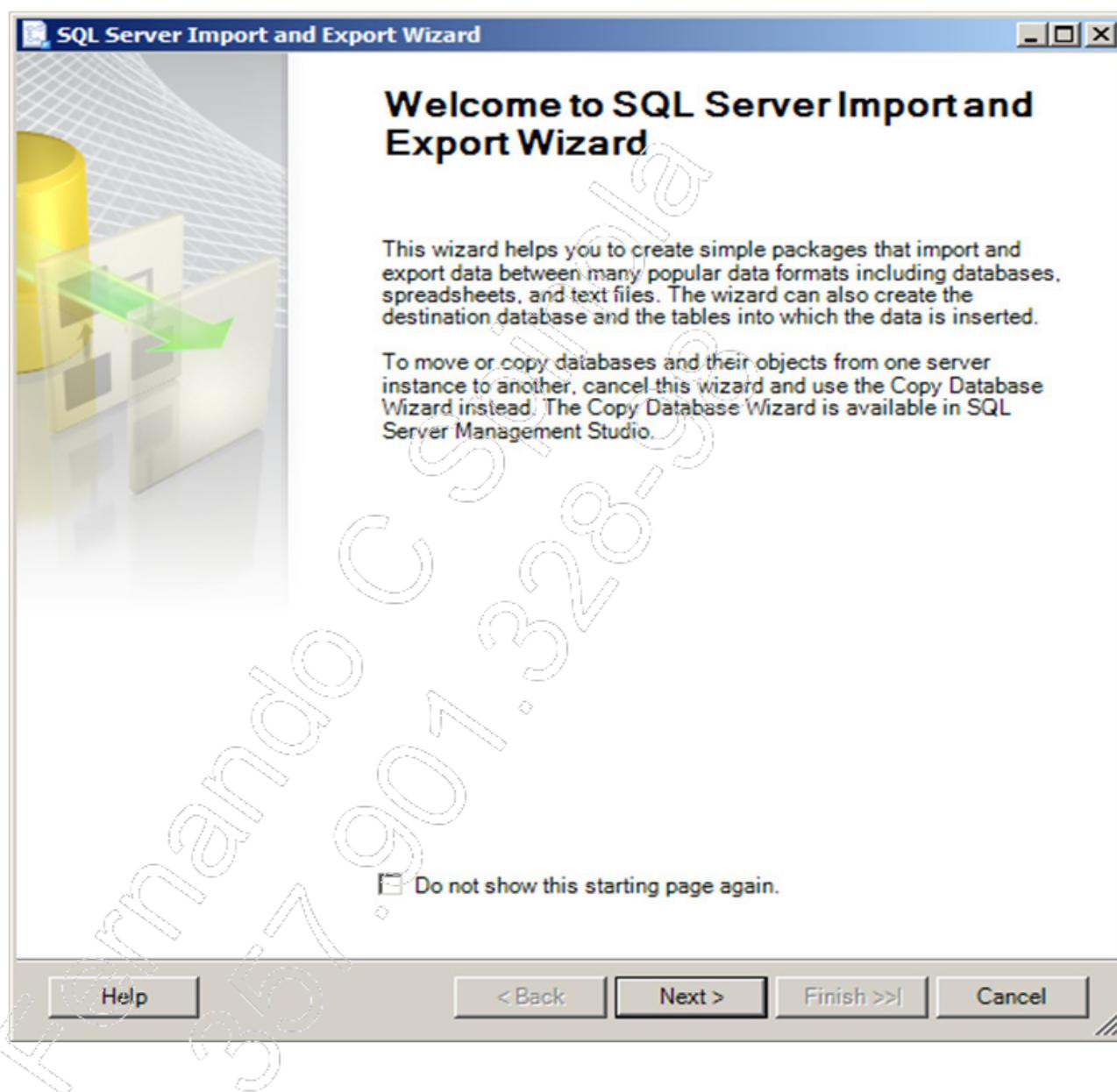
6.2. Exportando e importando dados

A exportação de dados é a técnica de backup lógico que permite extrair total ou parcialmente dados de uma ou mais tabelas em um banco de dados e gravar essas informações em diferentes tipos de arquivos de dados. O SQL Server possui duas ferramentas no SQL Server Management Studio (SSMS), também acessíveis no menu do SQL Server: uma para exportar dados e outra para a realização de importação de dados. Podemos exportar e importar de diversos tipos de arquivos, tais como Excel, Access, Oracle, texto e diversos provedores OLE DB. Isso pode ser feito inclusive entre bancos de dados SQL Server da mesma versão ou não.

A ferramenta de exportação e importação pode ser acessada através do menu, conforme a imagem a seguir:

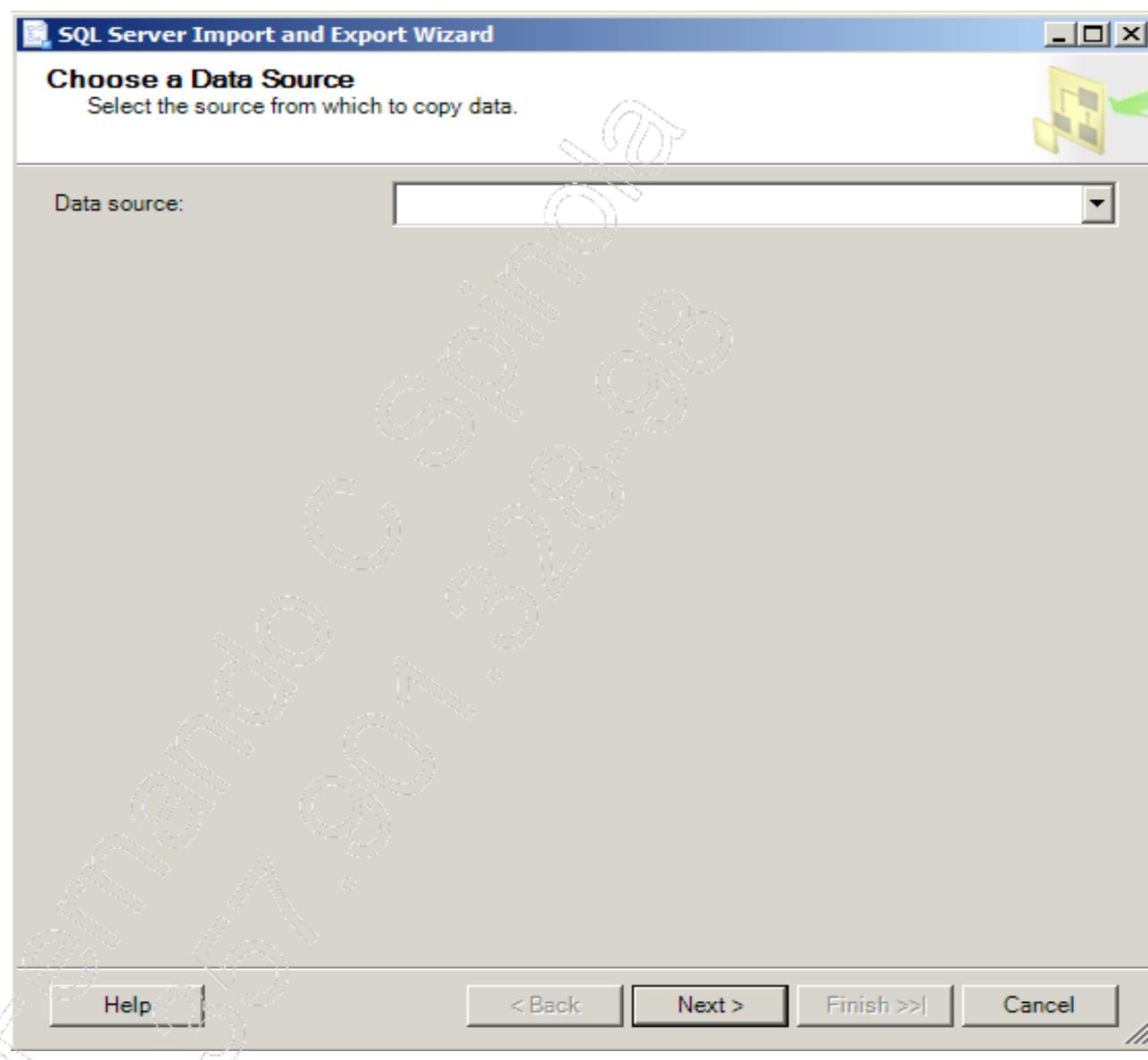


Selecione a opção **Export Data...** e clique em **Next** na tela que será exibida:



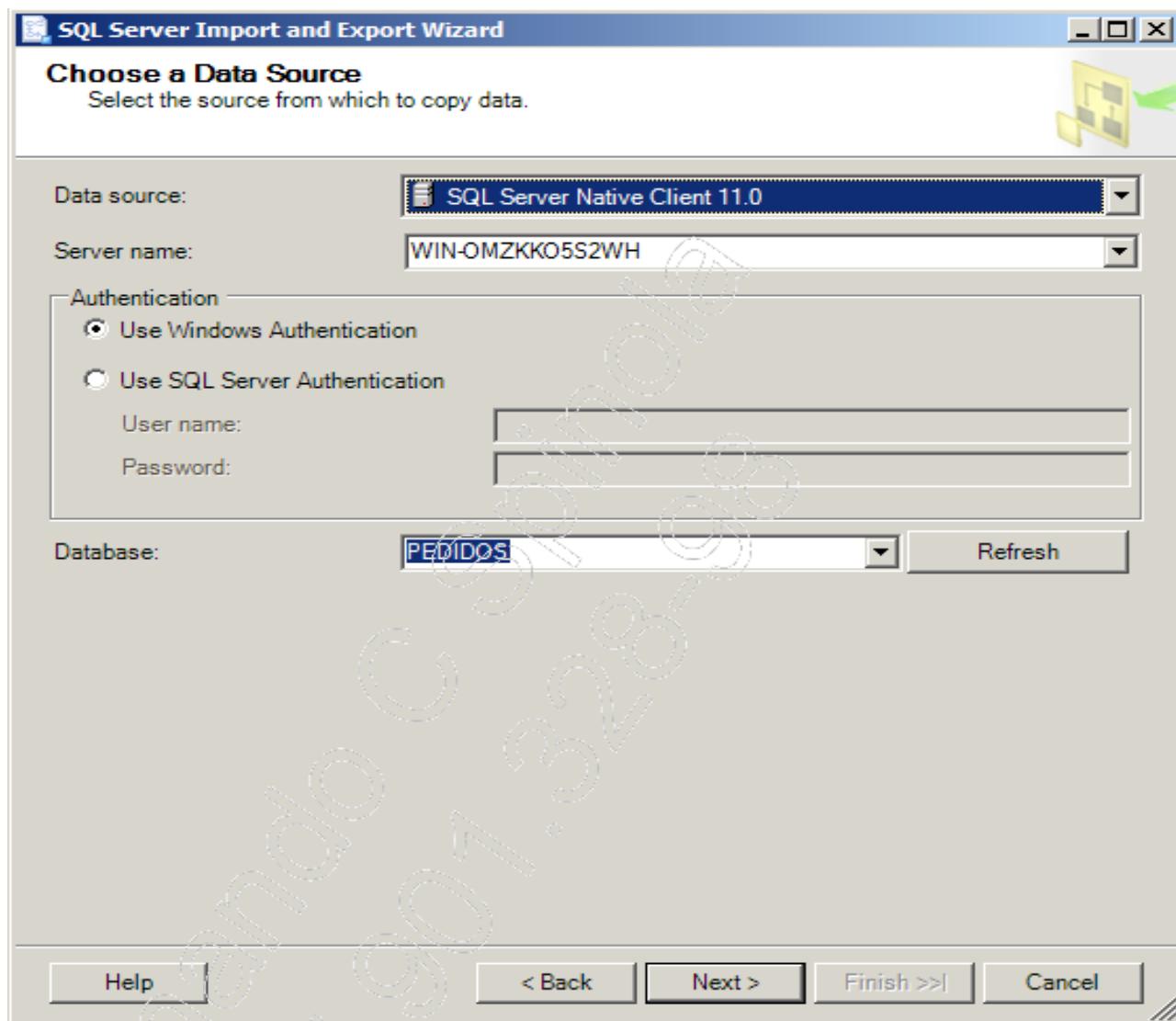
SQL 2014 - Módulo III

Em seguida, selecione a origem dos dados (**Data source**) a serem exportados. Clique em **Next** para prosseguir.



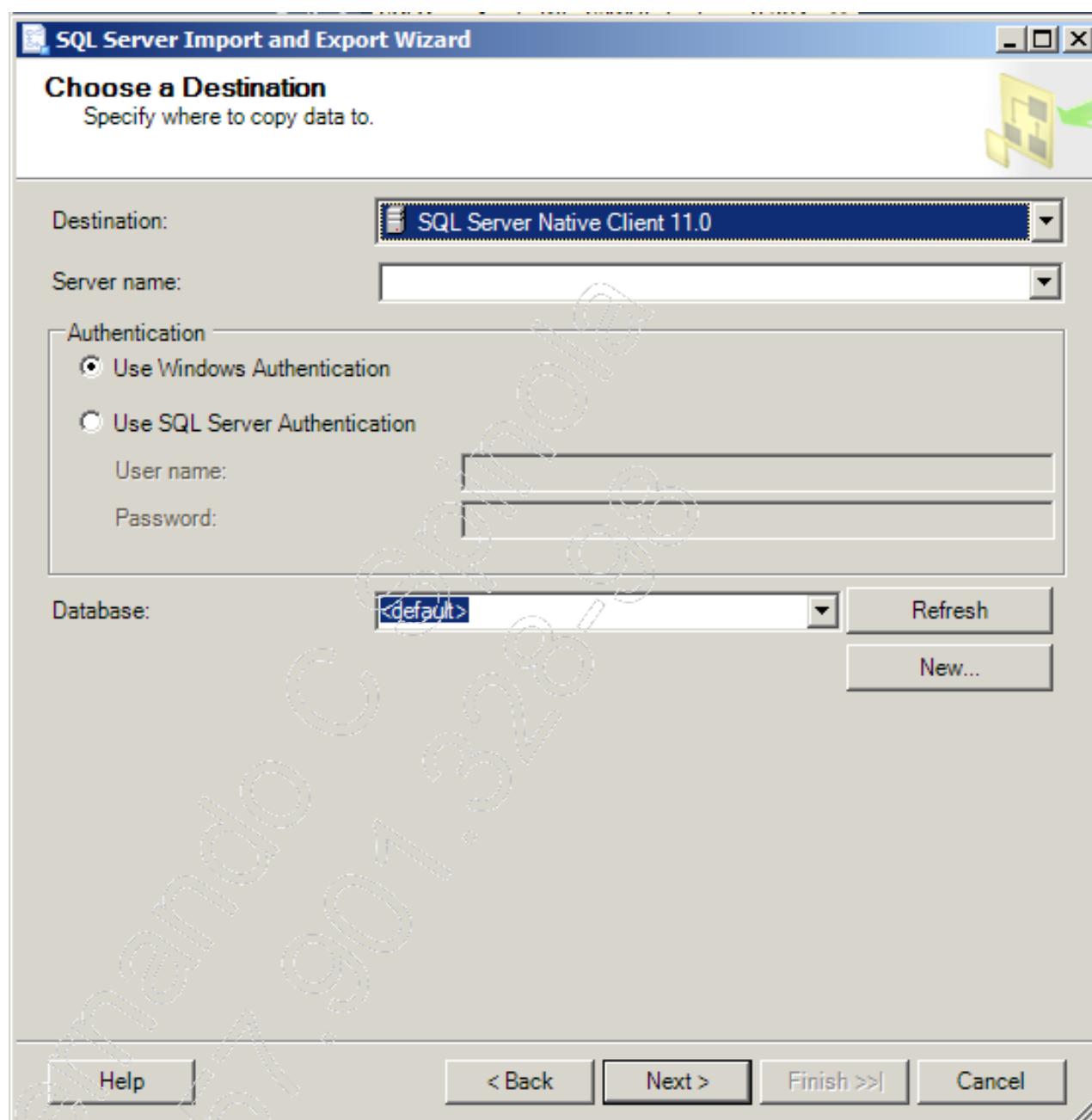
Transferência e manipulação de dados

Informe a origem na opção (**Data Source**) e o nome do servidor que deseja exportar (**Server Name**). Clique em **Next** para prosseguir.



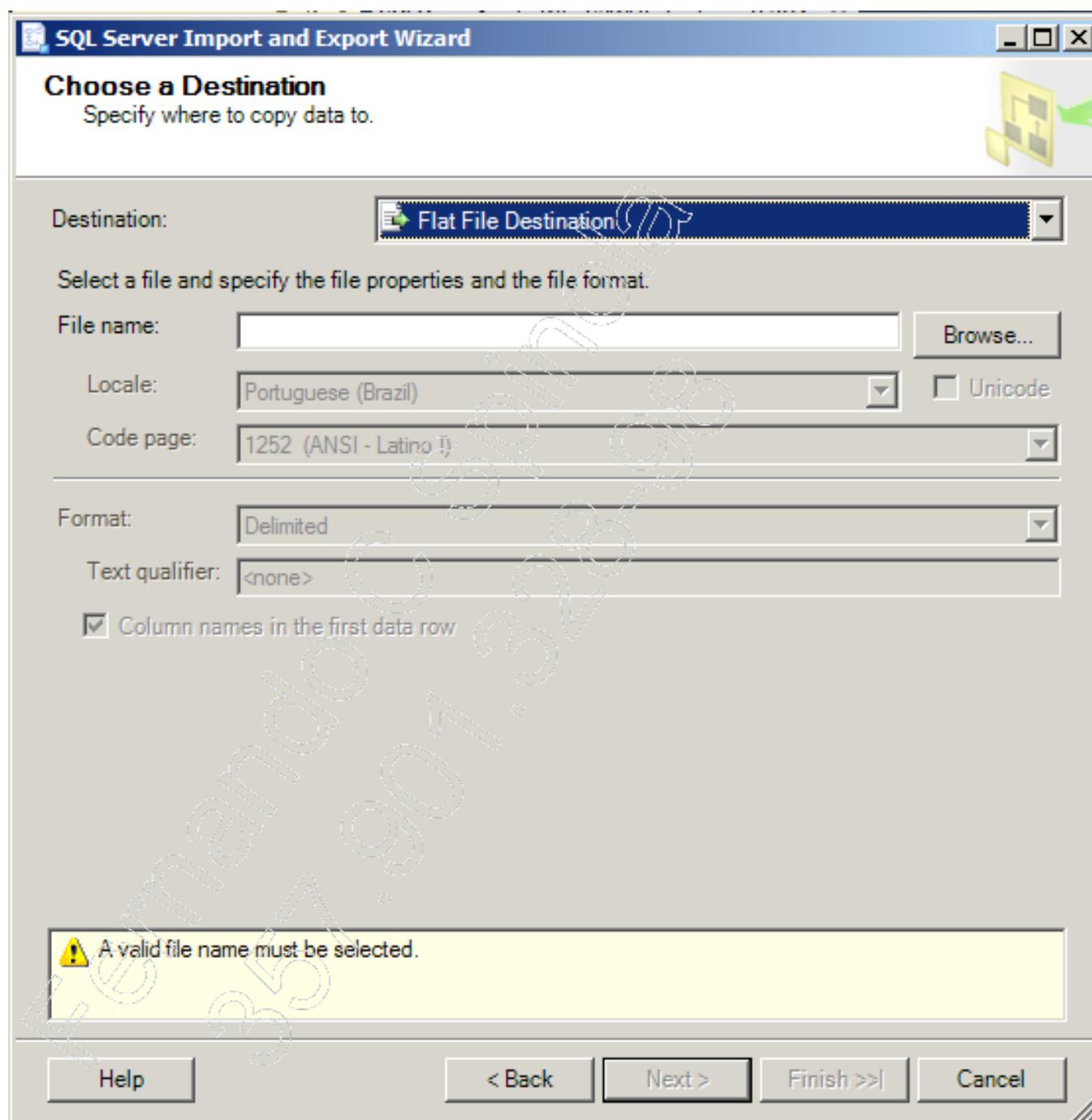
Caso a origem seja SQL Server, informe o modo de autenticação (**Authentication**), se será feita com o usuário conectado do Windows (**Use Windows Authentication**) ou com um usuário do SQL Server. Neste caso, deve-se informar o nome e a senha desse usuário. Além disso, o banco de dados (**Database**) que se deseja exportar também deve ser informado.

A ferramenta de exportação solicita que informemos seu destino. Para exportação entre SQL Server, deve-se informar o destino (**Destination**) e o nome do servidor (**Server name**):

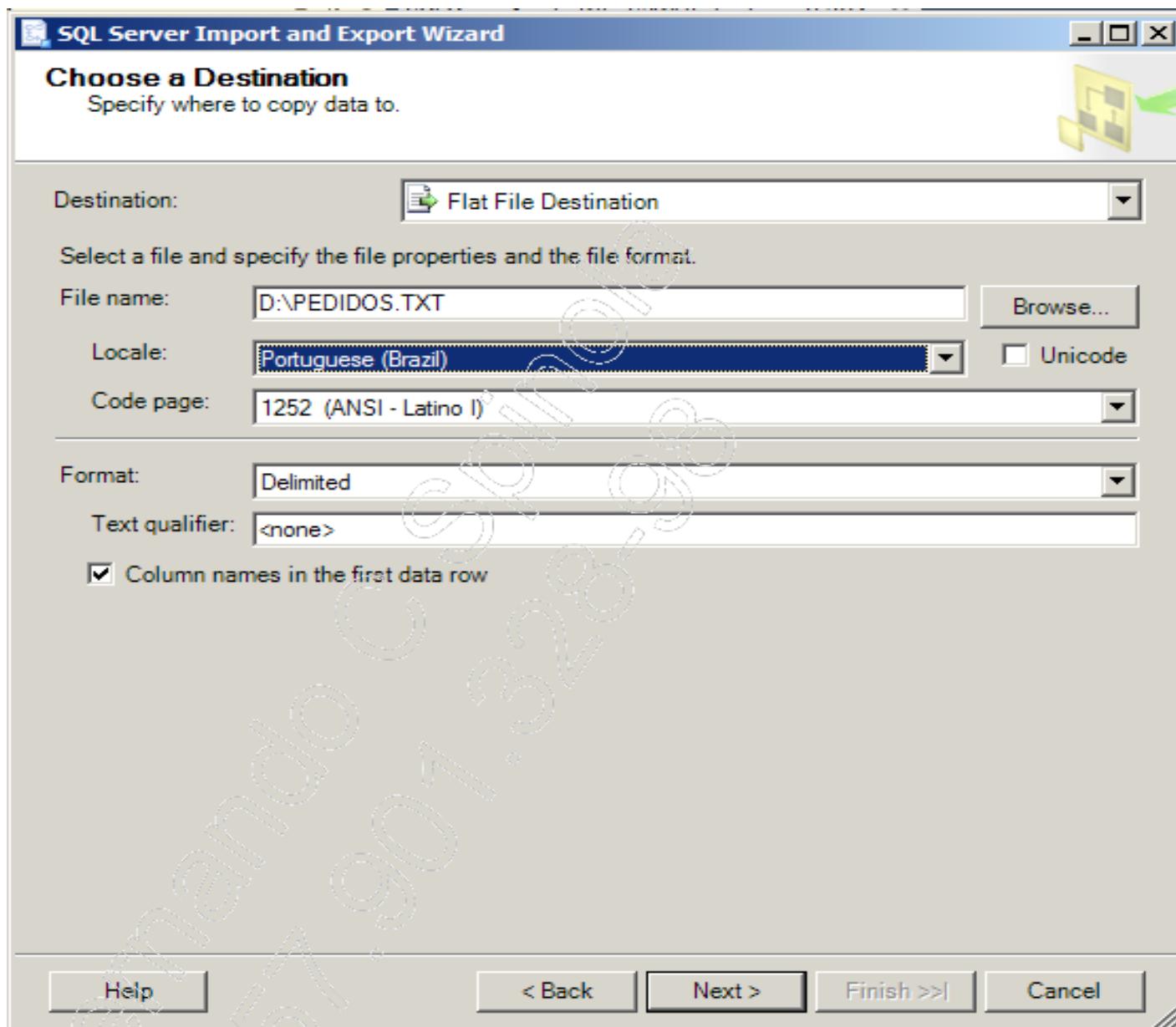


Caso o destino seja SQL Server, indique a forma de autenticação (**Authentication**), se será feita com o usuário conectado do Windows (**Use Windows Authentication**) ou com um usuário do SQL Server. Neste caso, deve-se informar o nome e a senha do usuário. Além disso, o banco de dados (**Database**) que se deseja exportar também deve ser informado. Clique em **Next** para prosseguir.

Vejamos a seguir a importação em arquivos texto (**Flat File**), Access, Excel e SQL Server:

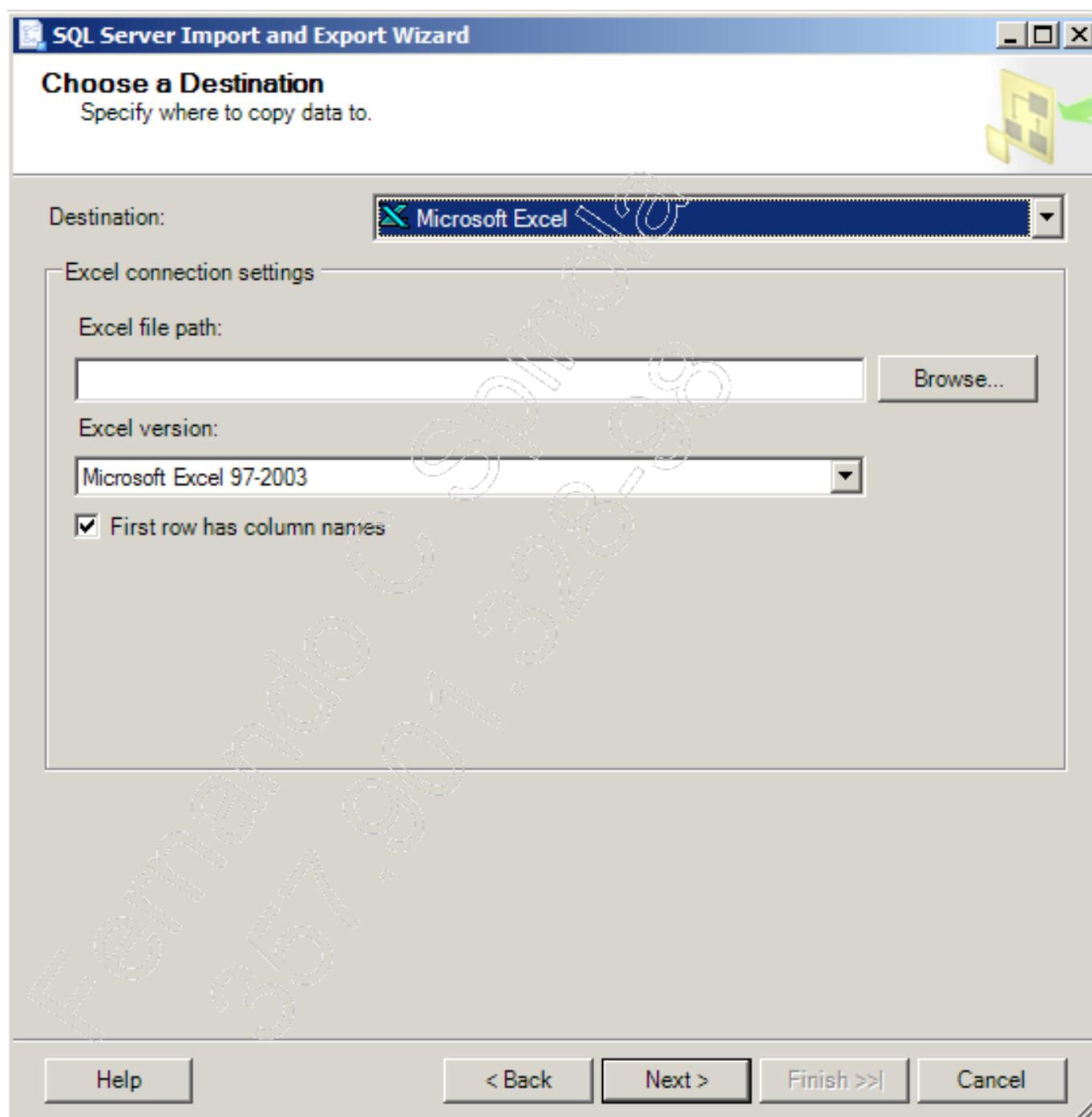


Na geração de arquivo texto (**Flat File**), indique o nome do arquivo que receberá os dados exportados na opção Nome do arquivo (**File Name**):

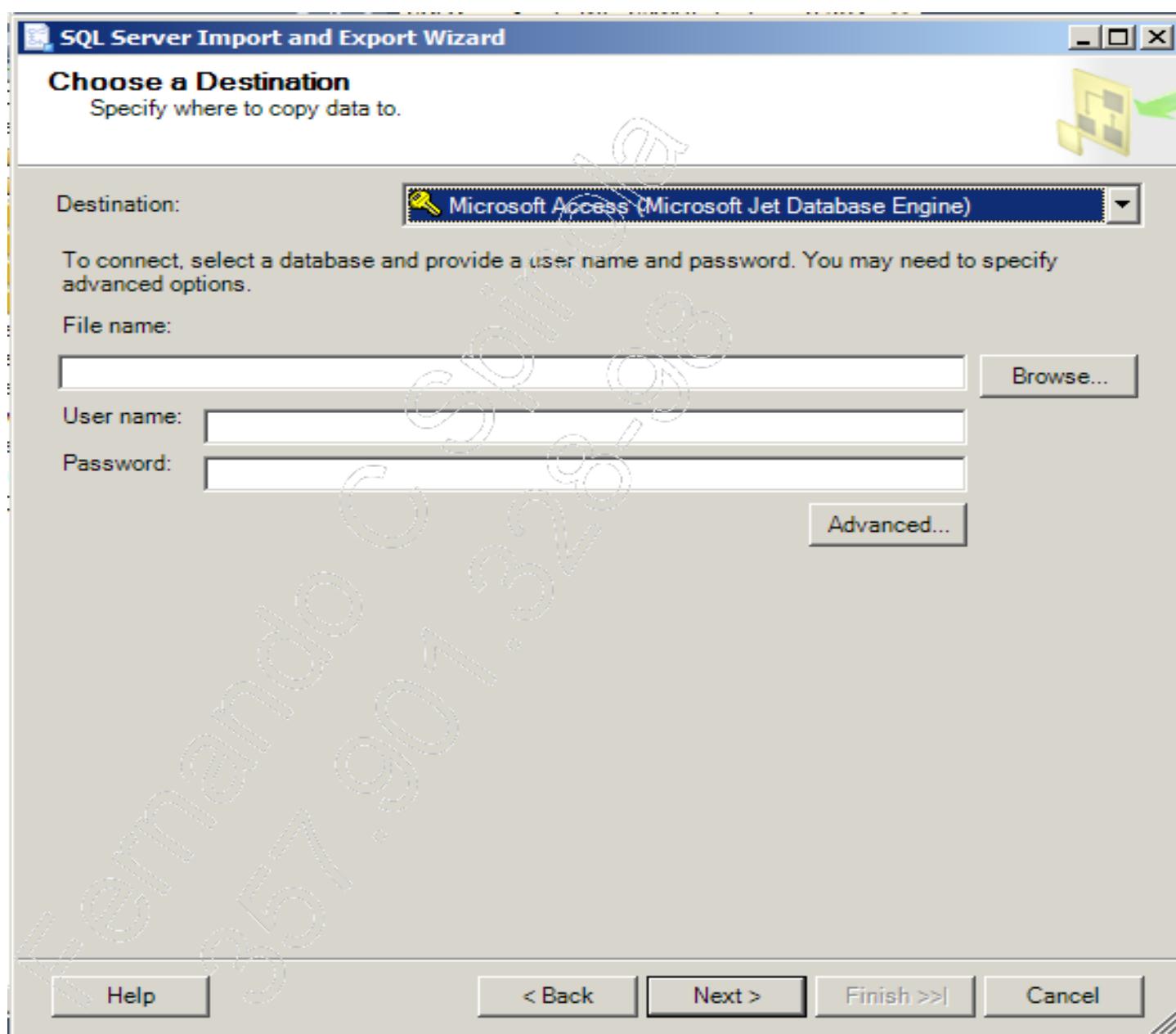


Informe o local (**Locale**), o tipo de página (**Code Page**), o tipo do formato (**Format**) e indique se os nomes das colunas deverão aparecer na primeira linha (**Column names in the first data row**). Em seguida, clique em **Next**.

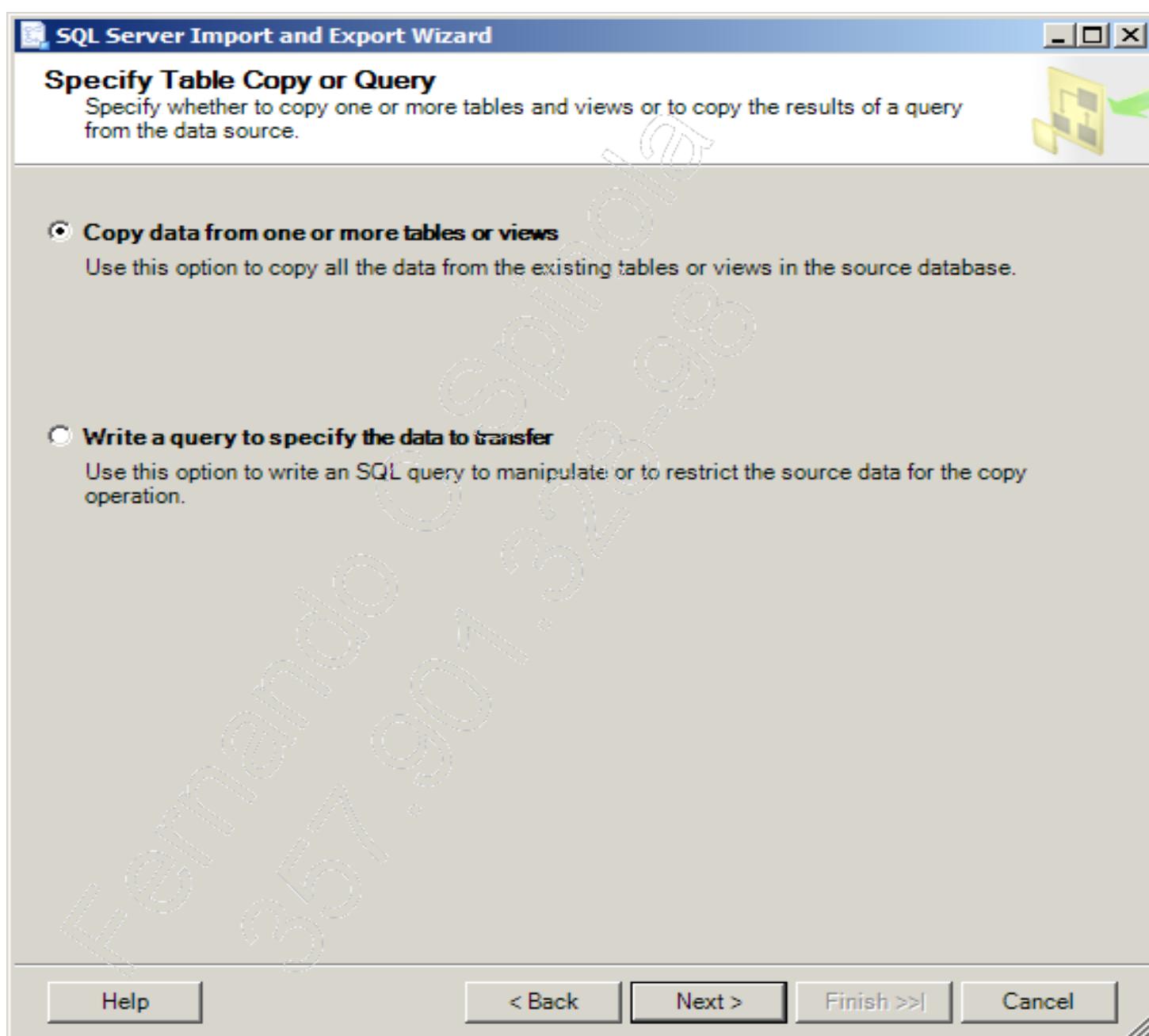
Para exportação no Excel, informe o nome do arquivo em **Excel file path** e a versão do Excel (**Excel version**). Indique ainda se a primeira linha do Excel deverá conter os nomes das colunas (**First row has column names**):



Para exportação para o Access, informe o nome do arquivo (**File Name**). Neste caso, o arquivo Access deve existir. Caso o arquivo tenha usuário e senha, esses dados devem ser informados:

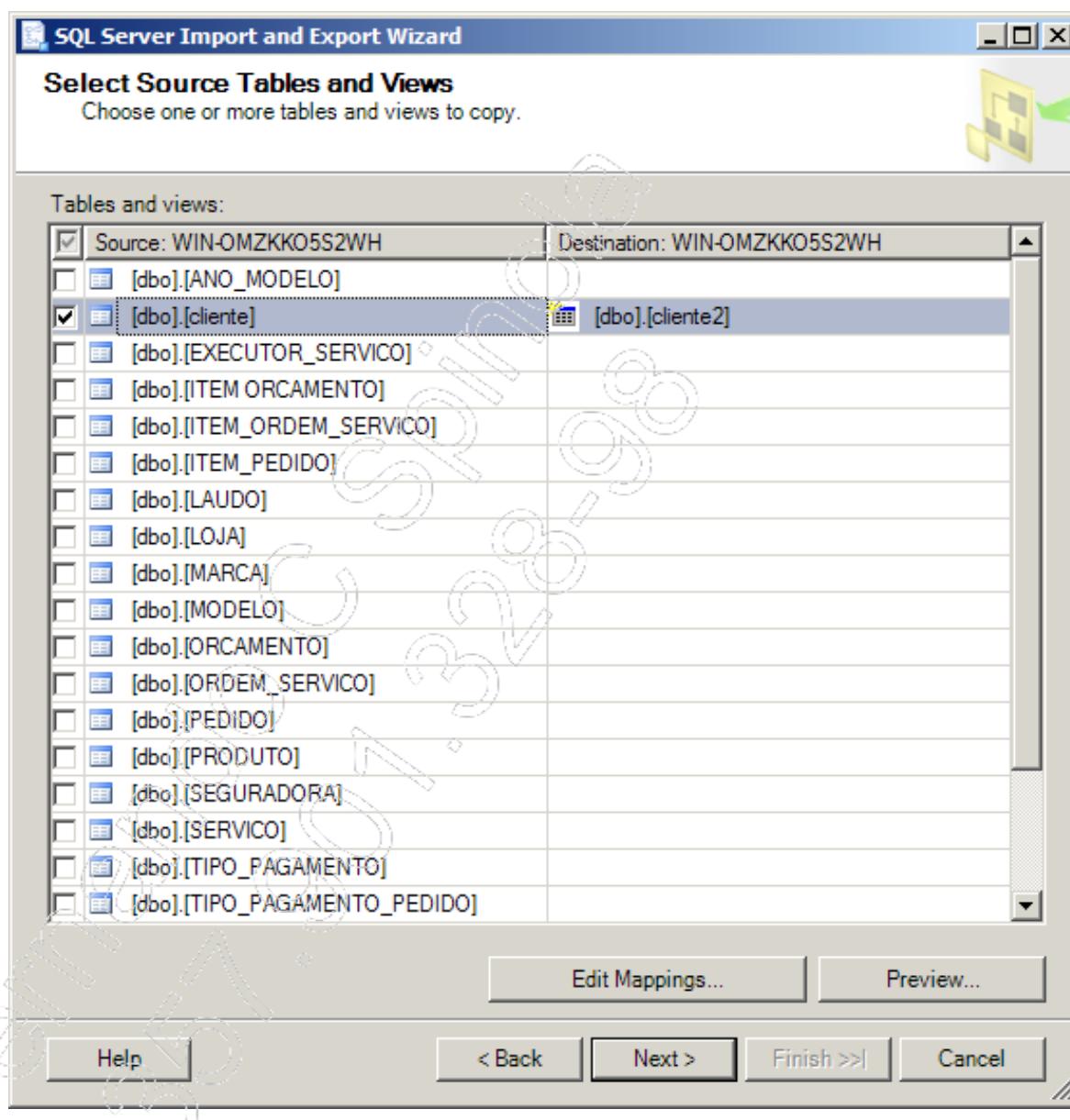


Uma vez selecionado o destino, clique em **Next**. O processo de exportação solicitará a indicação de qual(is) tabela(s) deverá(ão) ser exportada(s). Indique se deseja copiar todos os dados de uma ou mais tabelas ou se deseja escrever uma consulta como base para a exportação.

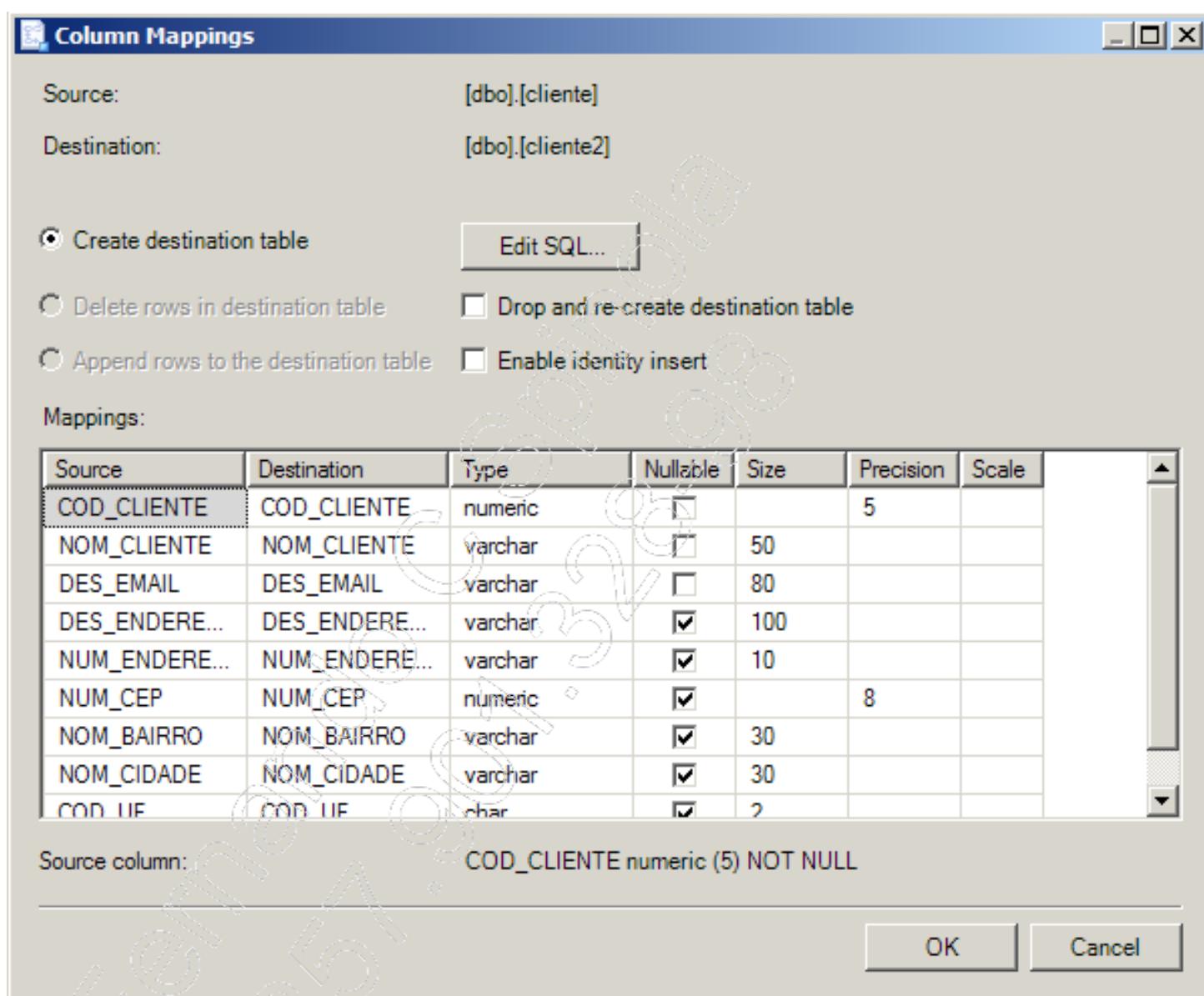


SQL 2014 - Módulo III

Clique em **Next** para prosseguir.



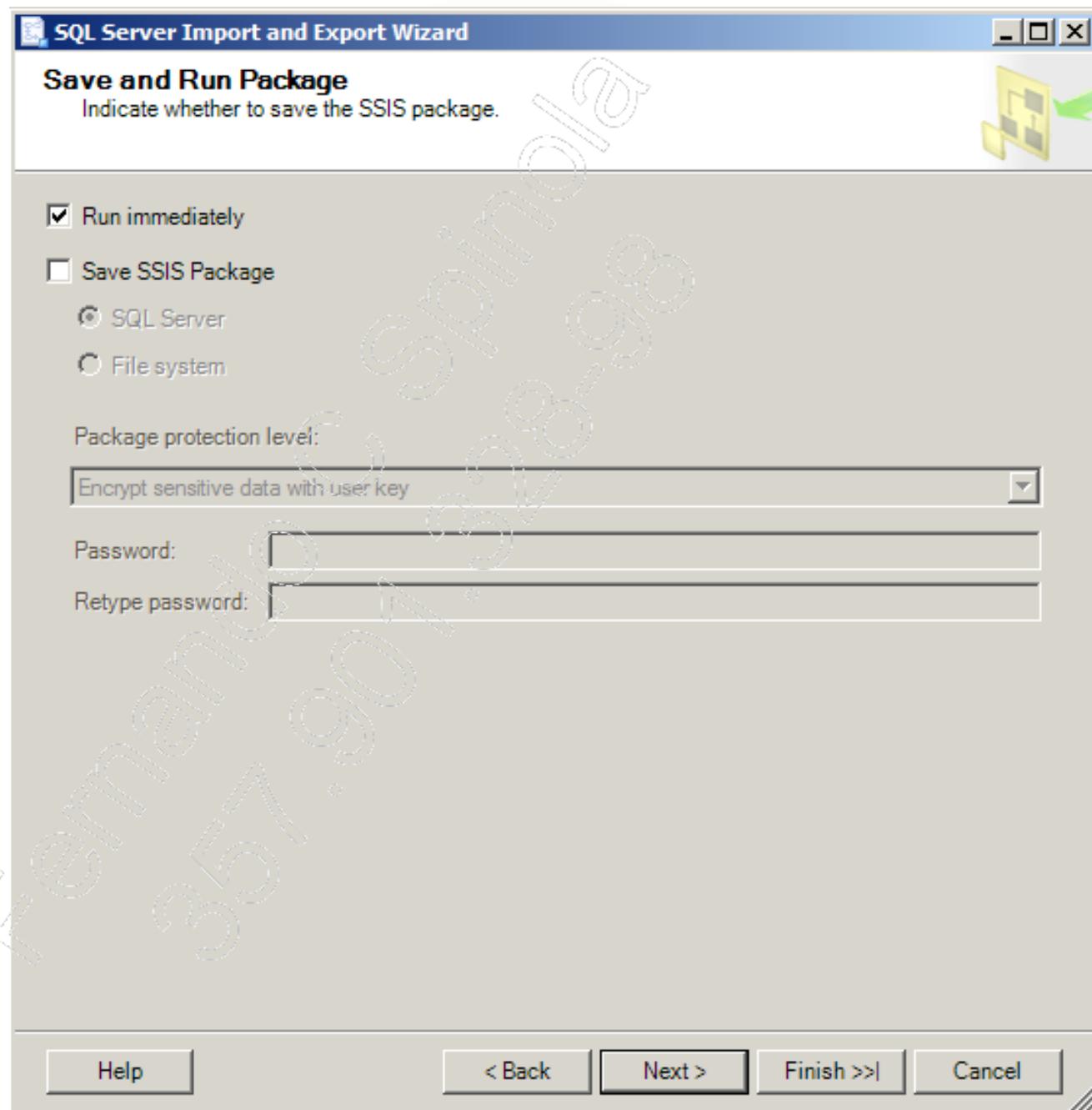
Selecione as tabelas a serem exportadas (**Source**) e o local para o qual serão importadas (**Destination**). É possível visualizar os dados que serão exportados usando o botão **Preview...**, além disso, também pode-se selecionar as colunas usando a opção **Edit Mappings**.



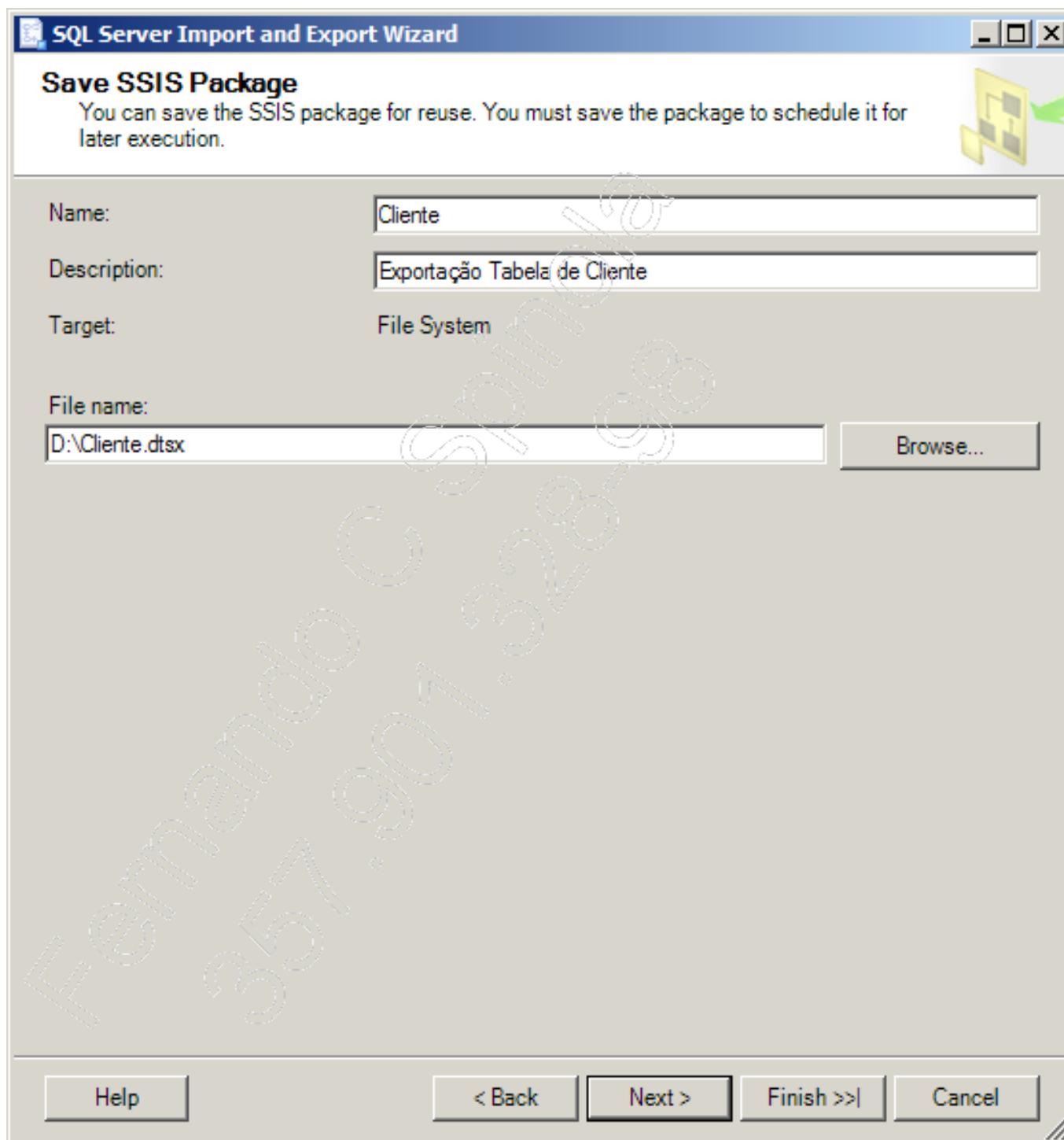
Podemos solicitar a eliminação e recriação da tabela caso ela exista (**Drop and re-create destination table**) e ainda habilitar a opção de inserir em campo que tenha a opção Identity (**Enable identity insert**).

Após clicar em **OK** na janela **Column Mapping** e em **Next** na janela **SQL Server Import and Export Wizard**, selecione as opções para execução:

- Execução imediata (**Run immediately**);
- Salvar em um pacote SSIS dentro do SQL Server ou em arquivo para posterior execução.

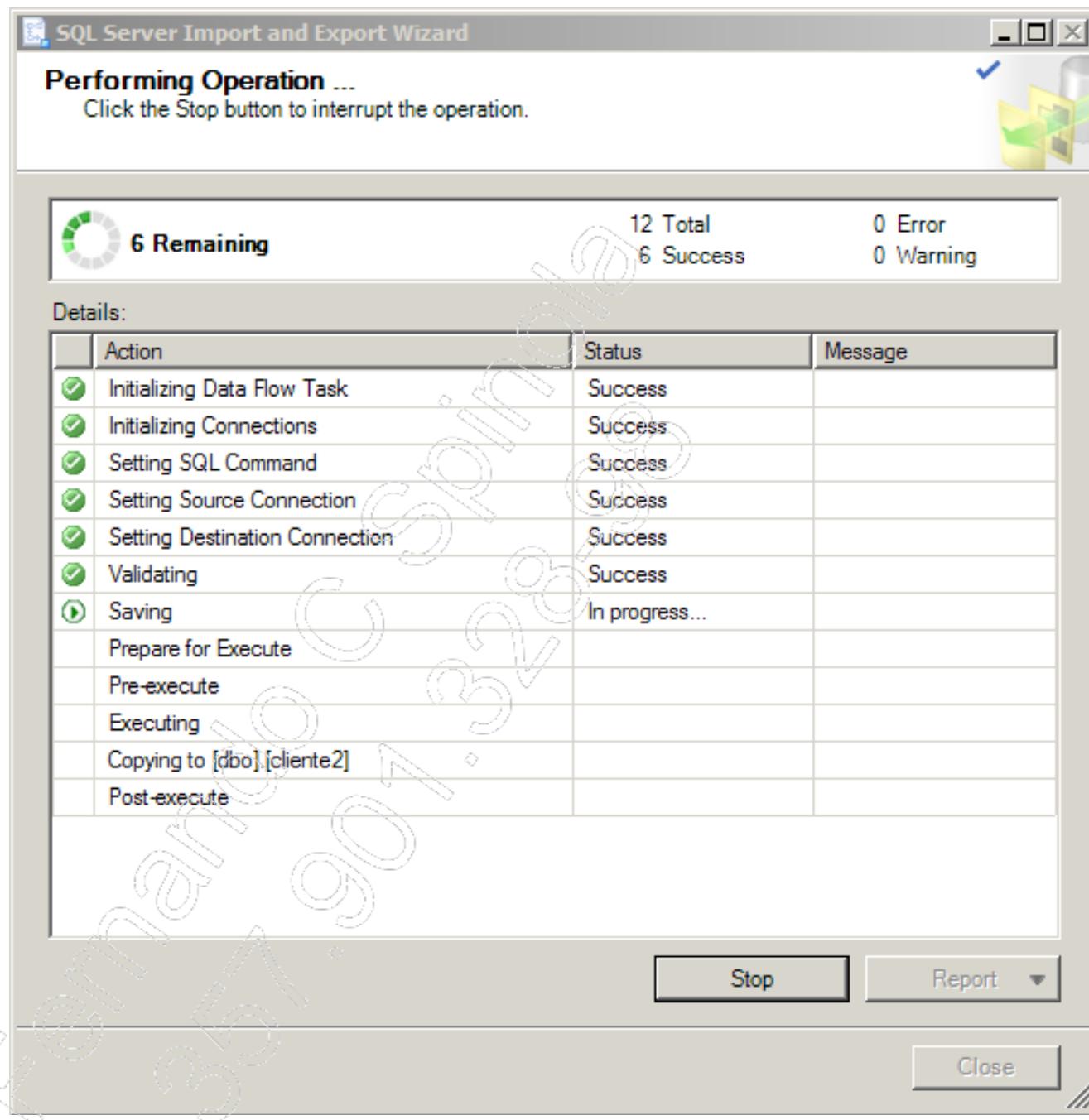


Após clicar em **Next**, informe o nome do pacote e o arquivo em que ele será gerado. A extensão padrão do arquivo é **dtsx**. Este arquivo poderá ser executado, inclusive através de linha de comando no próprio sistema operacional, dispensando assim a execução no modo gráfico.



SQL 2014 - Módulo III

Além disso, o arquivo **dtsx** poderá ser editado através da ferramenta **SQL Data Tools** (SQL Server 2014). Clique em **Finish** para prosseguir.



Ao final, realize a exportação e importação de dados.

6.3. Ferramenta de transformação e cópia de dados

O SQL Server 2014 dispõe de uma série de ferramentas para copiar e/ou transformar dados, úteis em processos de carga e transformação de dados vindos de arquivos ou de sistemas transacionais para serem utilizados em sistemas de tomada de decisão (DW - DataWarehouse / BI - Business Intelligence).

6.3.1. Bulk Copy Program (BCP)

Utilitário que permite realizar exportação e importação de dados no formato texto, recomendado para cargas de dados de pequenos a grandes volumes.

A seguir, temos a sintaxe do comando BCP:

```
bcp [database_name.] schema.{table_name | view_name | "query" {in  
data_file | out data_file | queryout data_file | format nul}  
  
[-a packet_size]  
[-b batch_size]  
[-c]  
[-C { ACP | OEM | RAW | code_page } ]  
[-d database_name]  
[-e err_file]  
[-E]  
[-f format_file]  
[-F first_row]  
[-h"hint [,...n]"]  
[-i input_file]  
[-k]  
[-K application_intent]  
[-L last_row]  
[-m max_errors]  
[-n]  
[-N]  
[-o output_file]  
[-P password]  
[-q]  
[-r row_term]  
[-R]  
[-S [server_name[\instance_name]]]  
[-t field_term]  
[-T]  
[-U login_id]  
[-v]  
[-V (80 | 90 | 100 )]  
[-w]  
[-x]  
/?
```

Vejamos as opções do comando BCP:

- **Database_Name**: Nome do banco de dados a ser importado ou exportado e que contém os dados de tabela ou uma visão específica;
- **Owner**: Proprietário ou schema que contém a tabela ou visão a ser manipulada;
- **Table_Name**: Nome da tabela que originará ou receberá os dados de um arquivo texto;
- **View_Name**: Nome da visão que originará ou receberá os dados de um arquivo texto;
- **Query**: Consulta Transact-SQL que retornará dados. A consulta deverá estar entre aspas duplas. Esta opção deve ser usada em conjunto com a opção queryout;
- **In|out|queryout|format**:
 - **In**: Indica que será uma importação de dados do arquivo texto para a tabela;
 - **Out**: Indica que será uma exportação de dados da tabela para o arquivo texto;
 - **Queryout**: Indica que uma consulta em uma ou mais tabelas ou visões serão exportadas para um arquivo texto;
 - **Format**: Esta opção deve ser utilizada em conjunto com a opção format_file.
- **Data_file**: Nome do arquivo de dados de origem ou destino, incluindo o diretório. O tamanho máximo é de 255 caracteres;
- **-m max_errors**: Limita o número máximo de erros durante a carga de importação e de exportação. O valor padrão é 10. Caso o número seja atingido, a carga será interrompida;

- **-f format_file:** Esta opção gera arquivos usando os valores **-c** (especifica que dados são caracteres), **-t** (caractere vírgula como separador de campo), **-x** (dados no formato xml), **-n** (especifica os tipos de dados nativos), **-f** (deve ser especificada caso a opção format seja utilizada), **-T** (deve utilizar conexão confiável; caso não utilize **-T**, escolha as opções **-U** e **-T**), **-w** (utilização de caracteres padrão Unicode. Usa o padrão **nchar**, o caractere **\t** para separação de campos e **\n** para separação de linhas), **-V(60|65|70|80|90|100)** para executar o **bcp** com tipos de dados de versões anteriores;
- **-e err_file:** Nome do arquivo de erros que registra as linhas que o **bcp** não conseguiu importar e ou exportar. Se esta opção não for utilizada em caso de erro, não serão registrados em nenhum lugar;
- **-F first_row:** Indica em qual linha do arquivo de dados deverá ser iniciada a leitura dos dados. Caso não seja especificado, o valor padrão é **1**;
- **-L Last_row:** Indica qual é a última linha que deverá ser transferida do arquivo de dados para o banco de dados. Caso não seja especificado, o valor padrão é **0**, o que indica que todas as linhas deverão ser transferidas;
- **-q:** Especifica o nome de uma tabela, visão, schema e banco de dados indicado entre aspas ou com espaço;
- **-t field terminator:** Caractere separador de colunas. O padrão é **\t**;
- **-r row terminator:** Caractere separador de registros. O padrão é **\n**;
- **-i input_file:** Arquivo em que as opções do comando **bcp** podem ser armazenadas para serem executadas, diminuindo assim a linha de comando;
- **-o output_file:** Nome do arquivo de saída do comando **bcp**;
- **-a packet_size:** Quantidade de bytes a serem transferidos de ou para o servidor. É recomendável o aumento deste parâmetro caso o arquivo lido ou gravado não esteja residente no servidor de banco de dados. Neste caso, haverá tráfego de rede;
- **-S server_name:** Nome da instância para conexão do comando **bcp**;

- **-U login_id:** Login_id do usuário que o bcp usará para conectar no banco de dados;
- **-P Password:** Senha correspondente ao Login_id;
- **-E:** Quando o arquivo importado contém valores de colunas do tipo IDENTITY.

6.3.2. Bulk Insert

O mecanismo de **Bulk Insert** permite a inserção de dados em massa vindos de arquivos texto. Possui maior eficiência que o mecanismo de BCP.

A seguir, temos a sintaxe do comando **Bulk Insert**:

```
BULK INSERT
    [ database_name . [ schema_name ] . | schema_name . ] [ table_
name | view_name ]
    FROM 'data_file'
    [ WITH
    (
        [ [ , ] BATCHSIZE = batch_size ]
        [ [ , ] CHECK_CONSTRAINTS ]
        [ [ , ] CODEPAGE = { 'ACP' | 'OEM' | 'RAW' | 'code_page' } ]
        [ [ , ] DATAFILETYPE =
            { 'char' | 'native' | 'widechar' | 'widenative' } ]
        [ [ , ] FIELDTERMINATOR = 'field_terminator' ]
        [ [ , ] FIRSTROW = first_row ]
        [ [ , ] FIRE_TRIGGERS ]
        [ [ , ] FORMATFILE = 'format_file_path' ]
        [ [ , ] KEEPIDENTITY ]
        [ [ , ] KEEPNULLS ]
        [ [ , ] KILOBYTES_PER_BATCH = kilobytes_per_batch ]
        [ [ , ] LASTROW = last_row ]
        [ [ , ] MAXERRORS = max_errors ]
        [ [ , ] ORDER ( { column [ ASC | DESC ] } [ ,...n ] ) ]
        [ [ , ] ROWS_PER_BATCH = rows_per_batch ]
        [ [ , ] ROWTERMINATOR = 'row_terminator' ]
        [ [ , ] TABLOCK ]
        [ [ , ] ERRORFILE = 'file_name' ]
    ) ]
```

Vejamos as opções a seguir:

- **Database_name**: Nome do banco de dados que sofrerá a carga de dados;
- **Schema_name**: Nome do schema que contém a tabela e ou visão. Caso o mesmo schema do usuário esteja realizando a operação, esse dado não precisa ser informado;
- **Table_name**: Nome da tabela ou visão do banco de dados que sofrerá o processo de carga de dados;
- **Data_file**: Nome do arquivo de dados que será lido para realização da carga, incluindo o drive, o diretório e subdiretórios, se houverem;
- **BATCHSIZE = [batch_size]**: Pode-se especificar a quantidade de linhas que será tratada como uma transação. Em caso de falhas ou de sucesso, este parâmetro indica o tamanho da transação;
- **CHECK CONSTRAINTS**: Indica que deverão ser tratadas as constraints da tabela que recebem os dados. Constraints são verificadas no momento do carregamento caso essa opção seja utilizada. O padrão é que as constraints são ignoradas no processo de carga;
- **CODE_PAGE**: Indica o tipo de código de página de dados;
- **DATAFILE_TYPE [= {'CHAR' | 'NATIVE' | 'WIDECHAR' | 'WIDENATIVE'}]**:
 - Caso o valor **char** seja utilizado para a opção, um arquivo de dados no formato caractere será copiado;
 - Caso o valor **native** seja utilizado, os tipos de dados nativos do banco para copiar os arquivos serão empregados;
 - Caso o valor seja **widechar** para a opção, um arquivo de dados no formato UNICODE será copiado;
 - Caso o valor seja **widenative** para a opção, será feita uma cópia seguindo os tipos de dados nativos, exceto **char**, **varchar** e **text**, que serão armazenados como caracteres padrão UNICODE.

- **ERRORFILE = ‘file_name’**: Especifica o arquivo de erro a ser gerado quando as transformações de dados não puderem ser realizadas. Caso o arquivo já exista depois de uma prévia execução, ocorrerá um erro de execução. Junto com a criação deste arquivo de erro, será gerado um arquivo com a extensão **.ERROR.txt** para referenciar cada linha e fornecer o diagnóstico dos erros, para serem corrigidos;
- **FIELDTERMINATOR [=‘field_terminator’]**: O valor padrão para esta opção é **\t**, porém, pode-se definir outro caractere separador de campos;
- **FIRST_ROW [= first_row]**: Com valor padrão 1, indica o número da primeira linha a ser lida;
- **FIRE_TRIGGERS**: Caso os gatilhos não sejam informados, eles não serão disparados no momento da carga de dados;
- **KEEPIDENTITY**: Caso especifique esta opção, o arquivo de dados conterá os valores da coluna do tipo **Identity**. Em caso de omissão, novos valores serão atribuídos a colunas do tipo **Identity**;
- **KEEPNULLS**: As colunas da tabela e/ou visão serão carregadas como nulas caso não sejam informadas no arquivo;
- **KILOBYTES_PER_BATCH [= kilobyte_per_batch]**: Quantidade em KB, correspondente a cada carga de dados realizada;
- **LASTROW [=last_row]**: Indica o número da última linha a ser processada. O valor padrão é 0 e isso indica que todas as linhas do arquivo serão processadas;
- **MAXERRORS [=max_errors]**: Número máximo de erros admitidos pelo processo de carga. O valor padrão é 0, o que indica que nenhum erro será admitido;

- **ORDER ({column [ASC|DESC]},[....n])**: Especifica a ordem dos arquivos de dados em caso de existirem mais do que um. Caso os dados do arquivo de dados estejam na ordem do índice clustered na tabela de destino, o processo de **Bulk Insert** terá uma grande melhoria de performance. Se os dados estiverem em outra ordem, a cláusula **order** será ignorada;
- **ROWTERMINATOR [=’row_terminator’]**: Indica o caractere terminador de registro. Caso não seja informado um caractere, \n é o valor padrão;
- **ROW_PER_BATCH**: Define a quantidade de linhas que deverão ser lidas por cada transação;
- **TABLOCK**: Define o travamento do tipo **TABLOCK** (travamento de toda a tabela) como tipo de travamento da tabela (**LOCK**).

 Processos de cargas tendem a ser mais eficientes quando o volume de leitura e manipulação é maior que um registro. Com isso, as transações se tornam maiores e o volume de I/O em disco (para leitura ou gravação de arquivo) e nos arquivos do banco de dados tende a ser menor.

Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo.

- A exportação e a importação de dados ocorrem quando tratamos de origem e fonte de um banco de dados, por exemplo, entre SQL Server e Oracle ou entre bases SQL Server. Isso ocorre no Excel, Flat Files e Access;
- A ferramenta assistente de exportação de dados pode ser invocada do próprio SSMS ou ainda externamente, através do menu de programas do SQL Server 2014;
- A manutenção do arquivo **dtsx** ou do pacote gravado no SQL Server pode ser feita pelo utilitário SQL Server Data Tools;
- A carga de dados realizada através de **BULK INSERT** é muito mais eficiente que a realizada através de **BCP**;
- Processos de cargas tendem a ser mais eficientes quando o volume de leitura e manipulação é maior que um registro. Com isso, as transações se tornam maiores e o volume de I/O em disco (para leitura ou gravação de arquivo) e nos arquivos do banco de dados tende a ser menor;
- Os mecanismos **BULK INSERT** e **BCP** possuem restrições de segurança que precisam ser avaliadas pela equipe de banco de dados e de segurança das empresas antes de serem utilizados.

6

Transferência e manipulação de dados

Teste seus conhecimentos

Fernanda
357.961-0200/1
Fernanda
357.961-0200/1



IMPACTA
EDITORA

1. Sobre BCP e BULK INSERT, qual afirmação está errada?

- a) BCP realiza exportação e importação de dados.
- b) BULK INSERT insere em bloco.
- c) BCP e BULK INSERT são antigos e não devem ser utilizados.
- d) BULK INSERT é mais rápido do que o INSERT.
- e) BCP pode abrir arquivos TXT e CSV.

2. Quais tipos de arquivo são suportados na exportação e importação do SQL?

- a) Texto, Excel, Access.
- b) Somente texto.
- c) Excel e Access.
- d) Texto, Excel, Access e provedores OLEDB.
- e) Texto e Access.

3. Qual é o utilitário que devemos usar para realizar a manutenção de um arquivo dtsx?

- a) SSMS
- b) SQL Server Data Tools
- c) Profiler
- d) Tunning Advisor
- e) Query Analyser

4. Exportação de dados é um tipo de backup?

- a) Sim, é considerado um backup lógico de caráter inconsistente.
- b) Não, mas é consistente.
- c) Não, backup somente através das ferramentas do SQL.
- d) Não, pois é de caráter inconsistente.
- e) Sim, mas é lento.

5. Onde podemos gravar um dtsx?

- a) Não é possível gravar este tipo de arquivo.
- b) Somente em arquivo.
- c) Somente no banco de dados.
- d) Em arquivo ou no banco de dados, mas não é possível realizar alterações.
- e) Em arquivo ou no banco de dados.

Transferência e manipulação de dados

M  os   obra!

6

transferência de manipulação e dados

Mãos à obra!



IMPACTA
EDITORIA

Laboratório 1

A – Utilizando o executável bcp para obter dados que estão no formato texto e carregá-los em uma tabela do SQL Server

1. Clique no botão **Start**, selecione **All Programs**, **Microsoft SQL Server 2014** e escolha a opção **SQL Server Management Studio**;
2. Conecte-se com a autenticação do Windows;
3. Na barra de ferramentas, clique no botão **New Query**;
4. Na barra de menus, clique na opção **File**, em seguida, em **Open** e em **File**. Abra o **Script_01** do **Capítulo_06**. Conecte-se com a autenticação do Windows;
5. Este script cria o database **Farmacia** com duas tabelas, uma chamada **Cliente**, outra chamada **ControleCli**, e um trigger chamado **T_ControlaCli**. Pressione F5 para que o script seja executado;
6. Clique novamente em **New Query**, na barra de ferramentas;
7. Executando os comandos a seguir, verifique que as tabelas **Cliente** e **ControleCli** estão vazias:

```
Use Farmacia
go
SELECT * FROM Cliente
go
SELECT * FROM ControleCli
go
```

8. Minimize o **SQL Server Management Studio**;
9. Localize, na pasta **Capítulo_06**, o arquivo **Clients.txt** e coloque-o na raiz do disco **C:**;
10. Clique no botão **Start** e, em seguida, em **Run**;
11. Na janela que será exibida, escreva **cmd** e clique em **OK**;

Transferência e manipulação de dados

12. No **Command Prompt**, digite e execute o comando a seguir:

```
bcp Farmacia.dbo.Cliente in "C:\Clientes.txt" -T -c -t"|" -E
```

13. Observe que 52 linhas de dados são copiadas pelo executável **bcp**;

14. Maximize o **SQL Server Management Studio**;

15. Executando os comandos a seguir, verifique que a tabela **Cliente** agora possui 52 registros e que a tabela **ControleCli** ainda está vazia:

```
Use Farmacia
go
SELECT * FROM Cliente
go
SELECT * FROM ControleCli
go
```

16. Vamos agora excluir os dados da tabela **Cliente** e, em seguida, vamos executar novamente o **bcp** com o parâmetro **-hFIRE_TRIGGERS**, para que o trigger **T_ControleCli** seja acionado e a tabela **ControleCli** também receba dados. Para tanto, primeiro devemos executar o comando a seguir:

```
TRUNCATE TABLE Cliente
```

17. No **Command Prompt**, escreva e execute o comando a seguir:

```
bcp Farmacia.dbo.Cliente in "C:\Clientes.txt" -T -c -t"|" -E
-hFIRE_TRIGGERS
```

18. Executando os comandos a seguir, verifique que a tabela **Cliente** possui 52 registros e que a tabela **ControleCli** também recebeu os 52 registros que foram inseridos em **Cliente**:

```
Use Farmacia
go
SELECT * FROM Cliente
go
SELECT * FROM ControleCli
go
```

Laboratório 2

A – Utilizando o comando BULK INSERT para obter dados que estão no formato texto e carregá-los em uma tabela do SQL Server

1. Clique no botão **Start**, em seguida, em **All Programs**, depois em **Microsoft SQL Server 2014** e escolha a opção **SQL Server Management Studio**;
2. Conecte-se com a autenticação do Windows;
3. Na barra de ferramentas, clique no botão **New Query**;
4. Na barra de menus, clique na opção **File**, em seguida, em **Open** e em **File** novamente. Abra o **Script_02** do **Capítulo_06**;
5. Este script cria uma tabela chamada **Funcionario**, outra chamada **Premio** e um trigger chamado **T_GeraPremio**. Utilize F5 para que o script seja executado;
6. Clique novamente em **New Query**, na barra de ferramentas. O **script_03** possui os comandos que serão utilizados adiante;
7. Executando os comandos a seguir, verifique que as tabelas **Funcionario** e **Premio** estão vazias:

```
Use Farmacia
go
SELECT * FROM Funcionario
go
SELECT * FROM Premio
go
```

8. Localize na pasta **Capítulo_06** o arquivo **Funcionarios.txt** e coloque-o na raiz do disco **C:**;
9. No **SQL Server Management Studio**, escreva e execute o comando a seguir:

```
BULK INSERT Funcionario
FROM 'c:\Funcionarios.txt'
WITH(FIELDTERMINATOR = '|')
```

Transferência e manipulação de dados

10. Executando os comandos adiante, verifique que a tabela **Funcionario** agora possui 22 registros e que a tabela **Premio** ainda está vazia;

```
Use Farmacia
go
SELECT * FROM Funcionario
go
SELECT * FROM Premio
go
```

11. Exclua os dados da tabela **Funcionario** e, em seguida, execute novamente o **BULK INSERT** com outros parâmetros. Para excluir os dados da tabela **Funcionario**, escreva e execute o comando a seguir:

```
TRUNCATE TABLE Funcionario
```

12. Escreva e execute o seguinte comando:

```
BULK INSERT Funcionario
FROM 'c:\Funcionarios.txt'
WITH(FIELDTERMINATOR = '|',
      FIRSTROW = 5,
      FIRE_TRIGGERS,
      LASTROW = 7,
      KEEPIDENTITY)
```

13. Executando os comandos a seguir, verifique que a tabela **Funcionario** possui três registros e que a tabela **Premio** também recebeu os mesmos três registros que foram inseridos em **Funcionario**, pois, com o parâmetro **FILE_TRIGGER**, o trigger que gera os prêmios foi executado. Observe que, devido ao parâmetro **KEEPIDENTITY**, os códigos dos funcionários originais do arquivo de origem foram mantidos e que, por causa dos parâmetros **FIRSTROW** e **LASTROW**, apenas os registros solicitados foram obtidos do arquivo texto e inseridos na tabela **Funcionario**.

```
Use Farmacia
go
SELECT * FROM Funcionario
go
SELECT * FROM Premio
go
```

Laboratório 3

A - Executando uma cópia de um database criado no SQL Server 2014

1. Clique no botão **Start**, em seguida, selecione **All Programs, Microsoft SQL Server 2014** e escolha a opção **SQL Server Management Studio**;
2. Na tela que será exibida, no campo **Server name**, escolha uma instância do **SQL Server 2014**, que será a nossa primeira instância do **SQL Server 2014**. Devemos nos conectar através da autenticação do Windows. Clique em **Connect**;
3. Assim que a conexão for estabelecida, na barra de ferramentas do **Object Explorer**, clique na opção **Connect** e escolha a opção **Database Engine**;
4. Na tela que será exibida, na opção **Server name**, selecione o nome de outra instância do **SQL Server 2014**, que será a nossa segunda instância do **SQL Server 2014**, e clique em **Connect**;
5. Assim que a conexão for estabelecida, na barra de menu, escolha a opção **File**, clique em **Open** e, em seguida, em **File** novamente;
6. Abra o **Script_04** da pasta **Capítulo_06**;
7. Em seguida, com o **Script_04** aberto, pressione F5 para que ele seja executado;
8. Feche o **Script_04**;
9. Clique com o botão direito do mouse sobre a pasta **Databases** da segunda instância do **SQL Server 2014** e, em seguida, selecione a opção **Refresh** para que o database **Siscom** possa ser visualizado;
10. Clique com o botão direito do mouse sobre o database **Siscom**, escolha a opção **Tasks** e, em seguida, escolha a opção **Copy Database**;
11. Na tela que será exibida, clique no botão **Next**;
12. Na próxima tela, no campo **Source Server** deverá estar o nome da segunda instância do **SQL Server 2014**. Se não estiver, clique no botão com as reticências (...) e selecione a instância desejada;

Transferência e manipulação de dados

13. Selecione a opção **Use Windows Authentication** e clique em **Next**;
14. No campo **Destination Server** da próxima tela, clique nas reticências (...) e escolha o nome da primeira instância do **SQL Server 2014** que desejamos que seja o destino do database **Siscom**;
15. Selecione a opção **Use the SQL Server Management Object method** e clique na opção **Next**;
16. Na tela que será exibida, o database **Siscom** já deve estar selecionado. Se não estiver, selecione-o e clique na opção **Next**;
17. Na próxima tela, observe que, no campo **Destination database**, o nome do database **Siscom** já vem escrito. Isso indica que, na primeira instância do **SQL Server 2014**, será criado um database com o mesmo nome do banco origem. Aceite o default e clique em **Next**;
18. Na próxima tela, no campo **Package name**, escreva **copia2014** e clique em **Next**;
19. Na próxima tela, aceite o default e clique em **Next**;
20. Clique em **Finish**;
21. Observe o **SQL Server** executando as suas configurações;
22. Observe que a cópia termina com sucesso. Clique no botão **Close**;
23. Clique com o botão direito do mouse na pasta **Databases** da conexão da primeira instância do **SQL Server 2014**, no **Object Explorer**, e escolha a opção **Refresh**;
24. Observe que o database **Siscom** foi criado;
25. Expanda o database **Siscom** e a pasta **Tables**. Observe que as tabelas do **Siscom** original foram criadas na primeira instância do **SQL Server 2014** também;

26. Clique com o botão direito do mouse sobre qualquer uma das tabelas e escolha a opção **Select Top 1000 rows** para verificar que os dados também foram todos copiados;
27. Com qualquer uma das tabelas expandida, clique na pasta **Keys** e em **Indexes**, para ver que as chaves primárias e estrangeiras também foram todas copiadas de um banco para o outro;
28. Feche todas as pastas que foram abertas;
29. Feche todas as queries que foram abertas.

Laboratório 4

A – Copiando o database Northwind de um database Access para a instância 2014 do SQL Server

1. Clique no botão **Start**, em seguida, selecione **All Programs, Microsoft SQL Server 2014** e escolha a opção **SQL Server Management Studio**;
2. Na tela que será exibida, no campo **Server name**, escolha a instância do SQL Server 2014, conectando-se com a autenticação do Windows. Clique em **Connect**;
3. Assim que a conexão for estabelecida, expanda a pasta **Databases**, clique com o botão direito do mouse e escolha a opção **New Database**;
4. Na tela que será exibida, no campo **Database name**, escreva **Northwind** e clique no botão **OK**;
5. Do lado esquerdo da tela, no **Object Browse**, clique com o botão direito do mouse sobre o database **Northwind** (que foi criado anteriormente) e escolha a opção **Tasks**. Em seguida, escolha a opção **Import Data**;
6. Na tela que será exibida, no campo **Data Source**, escolha a opção **Microsoft Access**;
7. Em seguida, clique no botão **Browse**, selecione o database **BancoAccessNorthwind** da pasta **Capitulo_06** e clique no botão **Next**;

Transferência e manipulação de dados

8. Na próxima tela, observe que, no campo **Destination**, deverá estar selecionada a opção **SQL Server Native Client** e que, no campo **Database**, já deverá estar selecionado o nome do database **Northwind**, que é o banco que receberá os dados do **Access**. Aceite todas estas opções como default e clique em **Next**;
9. Na próxima tela, deixe selecionada a primeira opção e clique em **Next**;
10. Na próxima tela, que exibe o nome de todas as tabelas que serão copiadas, clique no botão **Select All** e em **Next**;
11. Aceite a opção default da próxima tela e clique em **Next**;
12. Clique em **Finish** duas vezes e observe o **SQL Server** copiando os dados de um banco para o outro;
13. Assim que tudo terminar, clique em **Close**;
14. No **Object Explorer** do **SQL Server Management Studio**, expanda o database **Northwind** e a pasta **Tables**, observando que todas as tabelas do database **Access** foram copiadas para o **SQL Server 2014**.

Laboratório 5

A – Copiando os dados de uma planilha do Excel para o database **Siscom** da instância do **SQL Server 2014**

1. Clique no botão **Start**, em **Programs**, em **Microsoft SQL Server 2014** e escolha a opção **SQL Server Management Studio**;
2. Na tela que será exibida, no campo **Server name**, escolha a instância do **SQL Server 2014** conectando-se com a autenticação do Windows. Depois, clique em **Connect**;
3. Assim que a conexão for estabelecida, expanda a pasta **Databases** e clique com o botão direito do mouse sobre o database **Siscom**;
4. Escolha a opção **Tasks** e, em seguida, escolha a opção **Import data**;
5. Na tela que será exibida, clique em **Next**;

SQL 2014 - Módulo III

6. Na tela que será exibida, no campo **Data source**, escolha **Microsoft Excel**. Em seguida, nessa mesma tela, clique em **Browse** e escolha o arquivo chamado **Tabela.xls**, que está na pasta **Capitulo_06**. Aceite as demais opções como default e clique em **Next**;
7. Na próxima tela, observe que, no campo **Destination**, já deve estar marcada a opção **SQL Server Native Client** e, no campo **Database**, também já deve estar marcado **Siscom**. Sendo assim, clique em **Next**;
8. Na próxima tela, escolha a primeira opção e clique em **Next**;
9. Na tela que será exibida, no campo **Destination**, aparece a seguinte opção: **[Siscom].[dbo].[Plan1\$]**. No lugar da palavra **Plan1\$**, escreva a palavra **Vendas**. Nesse campo, ficará escrito: **[Siscom].[dbo].[Vendas]**;
10. Clique em **Edit Mappings** e observe a tabela que será criada no database **Siscom** da instância do SQL Server 2014. Clique em **Next**;
11. Na próxima tela, aceite o default e clique em **Finish** duas vezes;
12. Observe o SQL Server executando as opções configuradas. Assim que terminar o processo, clique em **Close**;
13. Do lado esquerdo da tela, no **Object Explorer**, expanda a pasta **Databases**, clique com o botão direito do mouse sobre a pasta **Tables** e escolha a opção **Refresh**;
14. Abra uma nova query e execute o seguinte comando:

```
Use Siscom
go
SELECT * FROM Vendas
go
```

15. Verifique que 500 registros foram inseridos nesta tabela. Para encerrar, feche a query dos dados.

Segurança de dados

7

- ✓ Login e usuário;
- ✓ Principals e securables;
- ✓ Gerenciamento de acesso à instância;
- ✓ Gerenciamento de acesso aos bancos de dados;
- ✓ Grupos de permissões criados pelo usuário;
- ✓ Grupos de permissões criados para aplicações;
- ✓ Permissionamento;
- ✓ Schema;
- ✓ Credenciais.



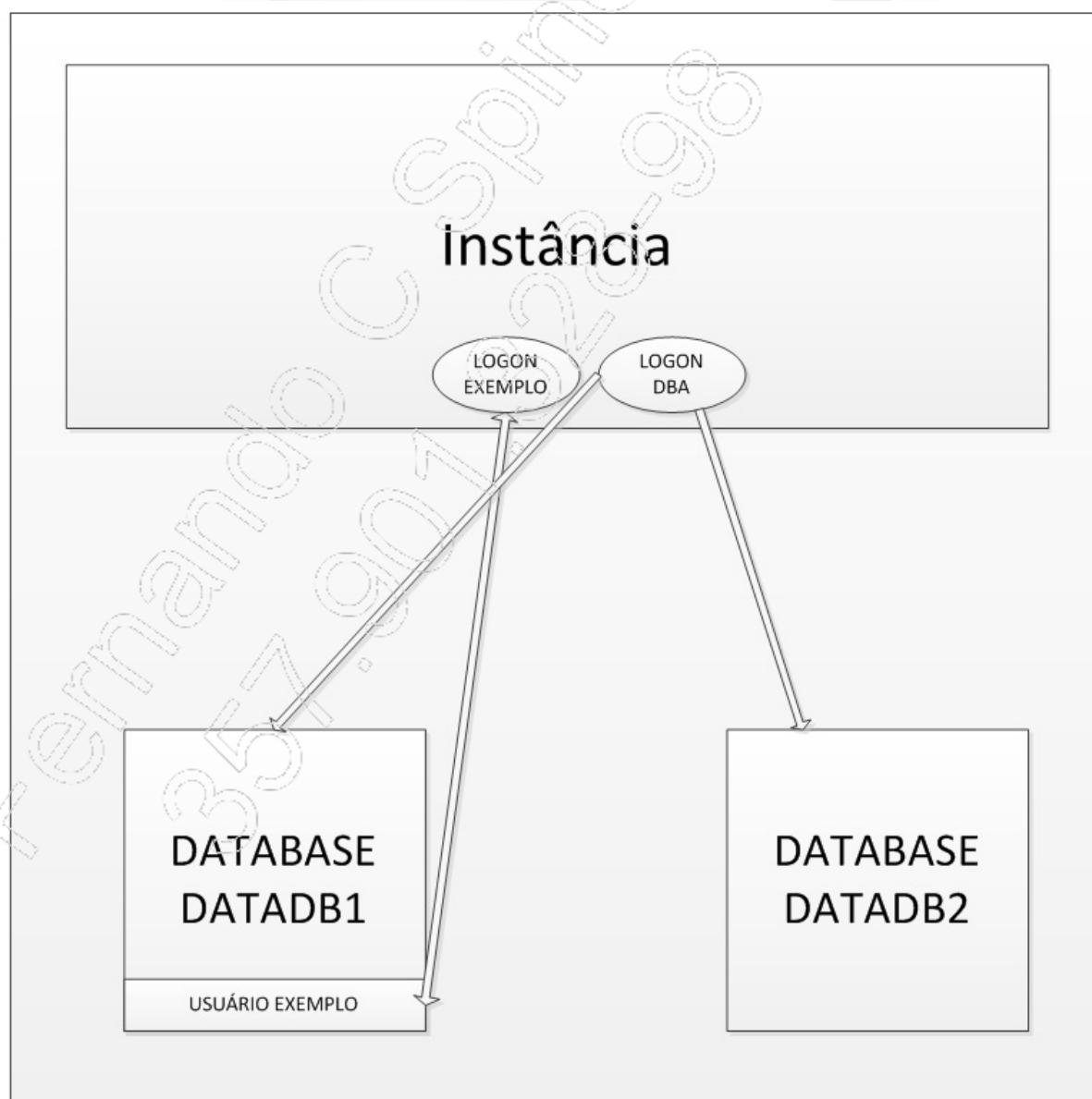
IMPACTA
EDITORA

7.1. Login e usuário

O processo de autenticação em uma instância SQL Server ocorre por meio de um objeto chamado **login**, que deve ser criado nessa instância.

Login se confunde com o conceito de usuário, normalmente utilizado em ferramentas de banco de dados, por razões funcionais. O banco de dados SQL Server possui um objeto chamado login, que tem a finalidade de realizar a conexão a uma instância. A partir desta é que será feito o acesso aos bancos de dados que a compõem. Para que o login possa realizar o acesso a um banco de dados específico, é necessário que este Login tenha permissão para isso. A permissão pode ser dada de forma ampla ou mais específica.

As permissões amplas são dadas a logins que estejam associados a pessoas ou aplicações que necessitem de um maior nível de permissão (por exemplo, administradores de banco de dados, operadores de backup etc.). Já as permissões mais específicas devem ser dadas para determinar os acessos aos bancos de dados de forma individual. Para acessarmos um banco de dados específico, caso não tenhamos permissões mais amplas, é necessário possuir uma estrutura chamada de **usuário**. Este usuário estará conectado a apenas um login e permitirá que, através do login na instância, um banco de dados específico seja acessado. Um login poderá ter um ou mais usuários (um para cada banco de dados acessado através de permissões específicas). Para permissões mais amplas, como no caso dos administradores de banco de dados, não há a necessidade de criação de usuário para cada banco de dados, apenas o login é o suficiente.



Na imagem anterior, ilustramos os logins **exemplo** e **dba**. O login **dba**, por ser utilizado pelo administrador, recebe uma permissão de instância chamada de **server role**. Com isso, não há necessidade de ter controle de acesso para cada banco de dados, o que não ocorre com o login **exemplo**, que precisa acessar apenas o banco de dados **DATADB1**. Com isso, este login precisa de um usuário neste banco de dados, o que normalmente tem o mesmo nome do login. Cada usuário em um banco de dados está associado a apenas um único login.

7.2. Principals e securables

Principals são objetos que podem receber permissões para acessar outro objeto particular de um banco de dados, podendo ser um usuário ou um grupo de segurança (roles), este pode ser atribuído a um ou mais usuários.

Existem três classes de principals no SQL Server:

- **Windows Principals:** Contas de usuários Windows autenticados via sistema operacional;
- **SQL Server Principals:** São logins criados e gerenciados pelo SQL Server inclusive no que se refere à sua autenticação;
- **Database Principals:** São usuários (**Users**), Grupos de Segurança (**Roles**) e Grupos de Segurança de Aplicações (**Application Roles**).

Securables são os objetos para os quais as permissões e o controle do nível de acesso podem ser dados. Existem três escopos de proteção:

- **Escopo de servidor (Server Scope)**: São os logins, HTTP Endpoints, notificação de eventos (event notifications) e bancos de dados. Estes objetos existem no nível da instância, ou seja, fora dos bancos de dados;
- **Escopo de banco de dados (Database Scope)**: São usuários (**users**), grupos de segurança (**roles**) e CLR Assemblies (objetos integrados com a arquitetura .NET) existentes em um banco de dados particular;
- **Escopo de esquema (Schema Scope)**: São objetos associados a um esquema, tais como tabelas, visões, stored procedures.

Quando projetamos a segurança do banco de dados, devemos trabalhar nestes três tipos de escopo. O mais operacional (dia a dia de um administrador de banco de dados) é o escopo de esquema, pois normalmente é o que apresenta maior nível de manipulação e ou acesso.

A autorização para uso dos Securables é dada pelos comandos SQL, denominados como DCL (Data Control Language), representados pelos comandos:

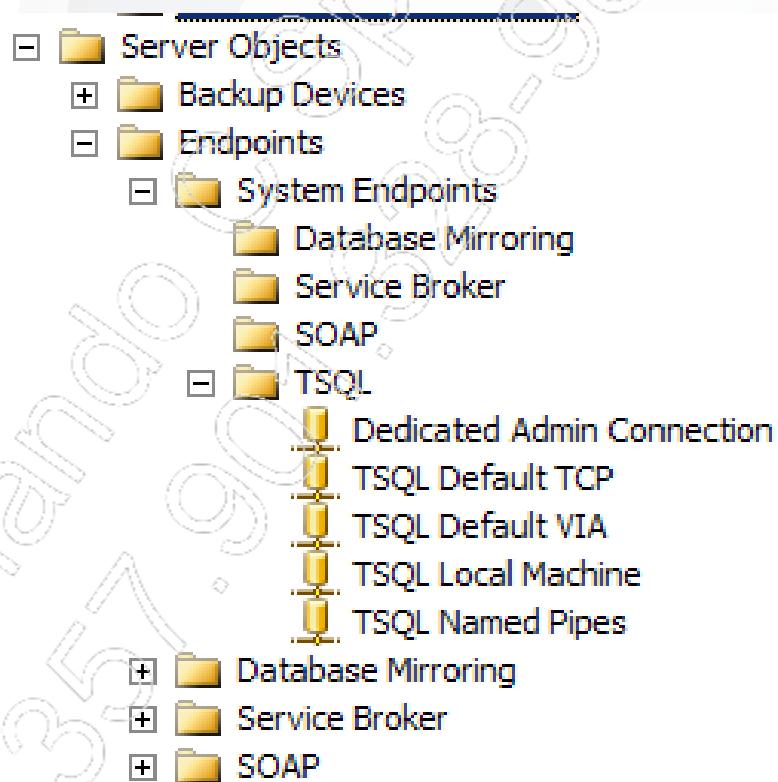
- **GRANT**: Comando que atribui direitos a um Principal. Através deste comando é que explicitamente atribuímos permissões a algum usuário;
- **REVOKE**: Comando que revoga uma permissão dada a um Principal. Somente as permissões dadas podem ser revogadas;
- **DENY**: Comando usado para evitar ou prevenir um acesso que possa ter sido dado anteriormente. Não é um comando padrão SQL ANSI.

ANSI é um órgão norte-americano para regulação e normatização de vários segmentos de mercado, incluindo software. Este órgão regulamenta, entre outras coisas, o padrão da linguagem SQL.

7.2.1. Server Securables – Endpoints

Todos os privilégios do tipo Server Securables podem ser atribuídos no nível da instância para Endpoints, um recurso utilizado para permitir o acesso remoto a um banco de dados (por exemplo, através da Internet). Endpoints têm a finalidade de ser um meio de conexão ao banco de dados. São utilizados por aplicações e por conexões remotas.

Existem os seguintes tipos de Endpoints: **System**, **Database Mirroring**, **Service Broker** e **SOAP**.



Podemos criar ENDPOINTS através do comando **CREATE ENDPOINT**. Confira a sintaxe desse comando:

```
CREATE ENDPOINT endPointName [ AUTHORIZATION login ]
[ STATE = { STARTED | STOPPED | DISABLED } ]
AS { TCP } (
    <protocol_specific_arguments>
)
FOR { TSQL | SERVICE_BROKER | DATABASE_MIRRORING } (
    <language_specific_arguments>
)

<AS TCP_protocol_specific_arguments> ::==
AS TCP (
    LISTENER_PORT = listenerPort
    [ [ , ] LISTENER_IP = ALL | ( 4-part-ip ) | ( "ip_address_v6" ) ]
)

<FOR SERVICE_BROKER_language_specific_arguments> ::==
FOR SERVICE_BROKER (
    [ AUTHENTICATION =
        WINDOWS [ { NTLM | KERBEROS | NEGOTIATE } ]
        | CERTIFICATE certificate_name
        | WINDOWS [ { NTLM | KERBEROS | NEGOTIATE } ] CERTIFICATE cer-
        tificate_name
        | CERTIFICATE certificate_name WINDOWS [ { NTLM | KERBEROS | NE-
        GOTIATE } ]
    ]
    [ [ , ] ENCRYPTION = { DISABLED | { { SUPPORTED | REQUIRED }
        [ ALGORITHM { RC4 | AES | AES RC4 | RC4 AES } ] }
    ]
    [ [ , ] MESSAGE_FORWARDING = { ENABLED | DISABLED } ]
    [ [ , ] MESSAGE_FORWARD_SIZE = forward_size ]
)

<FOR DATABASE_MIRRORING_language_specific_arguments> ::==
FOR DATABASE_MIRRORING (
    [ AUTHENTICATION =
        WINDOWS [ { NTLM | KERBEROS | NEGOTIATE } ]
        | CERTIFICATE certificate_name
        | WINDOWS [ { NTLM | KERBEROS | NEGOTIATE } ] CERTIFICATE cer-
        tificate_name
        | CERTIFICATE certificate_name WINDOWS [ { NTLM | KERBEROS | NE-
        GOTIATE } ]
    ]
    [ [ [ , ] ] ENCRYPTION = { DISABLED | { { SUPPORTED | REQUIRED }
        [ ALGORITHM { RC4 | AES | AES RC4 | RC4 AES } ] }
    ]
    [ , ] ROLE = { WITNESS | PARTNER | ALL }
)
```

SQL 2014 - Módulo III

Em que:

- **endPointName**: Nome do endpoint;
- **AUTHORIZATION LOGIN**: Nome associado ao LOGIN. Se não for especificado, quem chamar o ENDPOINT será o dono deste recurso. Para atribuir esta propriedade, quem chamar deverá ter permissão com caráter IMPERSONATE associada ao login que for especificado;
- **STATE = { STARTED | STOPPED | DISABLED }**: É o estado em que o ENDPOINT será criado. Caso não seja especificado, o padrão será STOPPED. STARTED indica que o ENDPOINT será inicializado após a criação, STOPPED define que ele deverá estar parado, já em DISABLED o ENDPOINT estará desabilitado para uso;
- **AS { TCP }**: É o protocolo de rede a ser utilizado. O padrão é TCP;
- **FOR { TSQL | SERVICE_BROKER | DATABASE_MIRRORING }**: Especifica o tipo de ENDPOINT a ser criado;
- **LISTENER_PORT = listenerPort**: Especifica a porta para atendimento (listener) que será utilizada para o Service Broker. O padrão é a porta 4022, mas é possível especificar qualquer porta entre 1024 e 32767;
- **LISTENER_IP = ALL | (4-part-ip) | ("ip_address_v6")**: Especifica o endereço TCP/IP para atendimento ao ENDPOINT. O padrão é all, o que indica que o atendimento poderá ocorrer a partir de qualquer endereço válido.

Exemplos de comandos para criação de endpoints:

```
CREATE ENDPOINT [ConexaoUsuario]
STATE = STARTED
AS TCP
    (LISTENER_PORT = 1680, LISTENER_IP =ALL)
FOR TSQL() ;
GO
```

Para associar o ENDPOINT a um usuário:

```
GRANT CONNECT ON ENDPOINT::[ ConexaoUsuario] TO [Impacta\DBA] ;
```

Para criar um ENDPOINT para o database Mirroring:

```
CREATE ENDPOINT endpoint_mirroring  
    STATE = STARTED AS TCP ( LISTENER_PORT = 7022 )  
  
FOR DATABASE_MIRRORING  
    ( AUTHENTICATION = WINDOWS KERBEROS,  
      ENCRYPTION = SUPPORTED,  
      ROLE=ALL );
```

Exemplo de ENDPOINT para HTTP (Exemplo Webservice):

```
CREATE ENDPOINT [Jogador_EP]  
    STATE=STARTED  
    AS HTTP  
    (  
        PATH=N'/Jogador'  
        , PORTS = (CLEAR)  
        , AUTHENTICATION = (INTEGRATED)  
        , SITE=N'localhost'  
        , CLEAR_PORT = 8000  
    )  
    FOR SOAP  
    (  
        WEBMETHOD 'ListaJogador'  
        ( NAME=N'[SSISEperiments].[dbo].[usp_SelecionarRegistros-  
Jogador]' )  
        , BATCHES=DISABLED  
        , WSDL=DEFAULT  
        , DATABASE=N'SSISEperiments'  
        , NAMESPACE=N'http://SSISEperiments/Jogadores'  
    )  
GO
```

Os seguintes privilégios podem ser dados a ENDPOINTS:

- **ALTER**: Habilita o login que tenha privilégios para alterar o ENDPOINT;
- **CONNECT**: Habilita a conexão utilizando o ENDPOINT. Por padrão, todos os logins têm direito de conectar com os ENDPOINTS;
- **CONTROL**: Permite repassar direitos para outros Logins;
- **TAKE OWNERSHIP**: Permite que a propriedade de ENDPOINT seja passada para outro Login;
- **VIEW DEFINITION**: Permite ver as configurações do Endpoint sem que se possa modificá-las.

7.2.2. Server Securables – Logins

LOGINS são objetos utilizados para conexão com a instância SQL Server, podendo estar ou não associados a usuários (USERS). As permissões que podemos dar a logins são:

- **ALTER**: Permissão para alteração de login, por exemplo, o login **Teste** tem permissão para alterar o login **Teste2**;
- **CONTROL**: Permissão total sobre o login (ALTER, CONTROL e IMPERSONATE);
- **IMPERSONATE**: Permissão para executar ou acessar objeto como outro Login;
- **VIEW DEFINITION**: Permissão para ver as propriedades do Login sem alterá-las.

Podemos criar **Logins** através do comando **CREATE LOGIN**. Vejamos a seguir a sintaxe desse comando:

```
CREATE LOGIN login_name { WITH <option_list1> | FROM <sources> }

<option_list1> ::=  
    PASSWORD = { 'password' | hashed_password HASHED } [ MUST_  
CHANGE ]  
    [ , <option_list2> [ ,... ] ]

<option_list2> ::=  
    SID = sid  
    | DEFAULT_DATABASE = database  
    | DEFAULT_LANGUAGE = language  
    | CHECK_EXPIRATION = { ON | OFF}  
    | CHECK_POLICY = { ON | OFF}  
    | CREDENTIAL = credential_name

<sources> ::=  
    WINDOWS [ WITH <windows_options>[ ,... ] ]  
    | CERTIFICATE certname  
    | ASYMMETRIC KEY asym_key_name

<windows_options> ::=  
    DEFAULT_DATABASE = database  
    | DEFAULT_LANGUAGE = language
```

Em que:

- **Login_name**: Nome do login;
- **PASSWORD='password'**: Aplica-se a logins com autenticação do SQL Server. Deve-se especificar uma senha, definida por letras maiúsculas e minúsculas com pelo menos 8 caracteres (numéricos e alfanuméricos). O limite para senha é de 128 caracteres. As senhas não podem ter aspas simples nem o nome do login;

- **MUST_CHANGE**: Aplica-se a logins com autenticação do SQL Server. Esta opção implica na troca de senha pelo usuário na primeira vez que for realizar uma conexão com o banco de dados;
- **CREDENTIAL = credential_name**: Alternativamente, pode-se nomear uma credencial para um login no SQL Server. Esta credencial deve ter sido previamente criada. Deve-se vincular a credencial a um login, com exceção do logon SA, que não pode ser utilizado para essa finalidade;
- **SID = sid**: Aplicável apenas a logins autenticados pelo SQL Server. Especifica o GUID de um novo login do SQL Server. Caso não seja especificado, será nomeado um GUID pelo SQL Server;
- **DEFAULT_DATABASE = database**: Especifica qual é o banco de dados padrão atribuído ao login. Caso não seja indicado, o banco de dados será o MASTER;
- **DEFAULT_LANGUAGE = language**: Especifica o idioma a ser utilizado pelo login. Caso não seja definido, o idioma a ser utilizado será o padrão da instância. Caso seja alterado o padrão do idioma da instância, o padrão aplicado ao login não será alterado;
- **CHECK_EXPIRATION = { ON | OFF }**: Aplica-se apenas a logins SQL Server. Indica se a política de expiração deve ser aplicada ao login criado. Por padrão, o valor definido é OFF;
- **CHECK_POLICY = { ON | OFF }**: Aplica-se apenas a logins do SQL Server. Especifica se a política de senhas aplicadas ao Windows deve ser aplicada ao login. Isso implica em aderir às regras aplicáveis às senhas dos usuários no sistema operacional.

7.2.3. Criando Logins

Observe os exemplos a seguir:

- **Exemplo de criação de LOGIN autenticado pelo Windows**

```
CREATE LOGIN [IMPACTA\DBA] FROM WINDOWS;  
GO
```

- **Exemplo de criação de LOGIN autenticado pelo SQL Server**

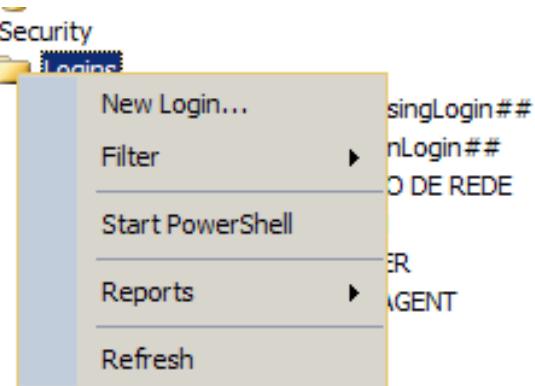
```
CREATE LOGIN DBA_IMPACTA WITH PASSWORD = 'DB@Imp@ct@!'  
GO
```



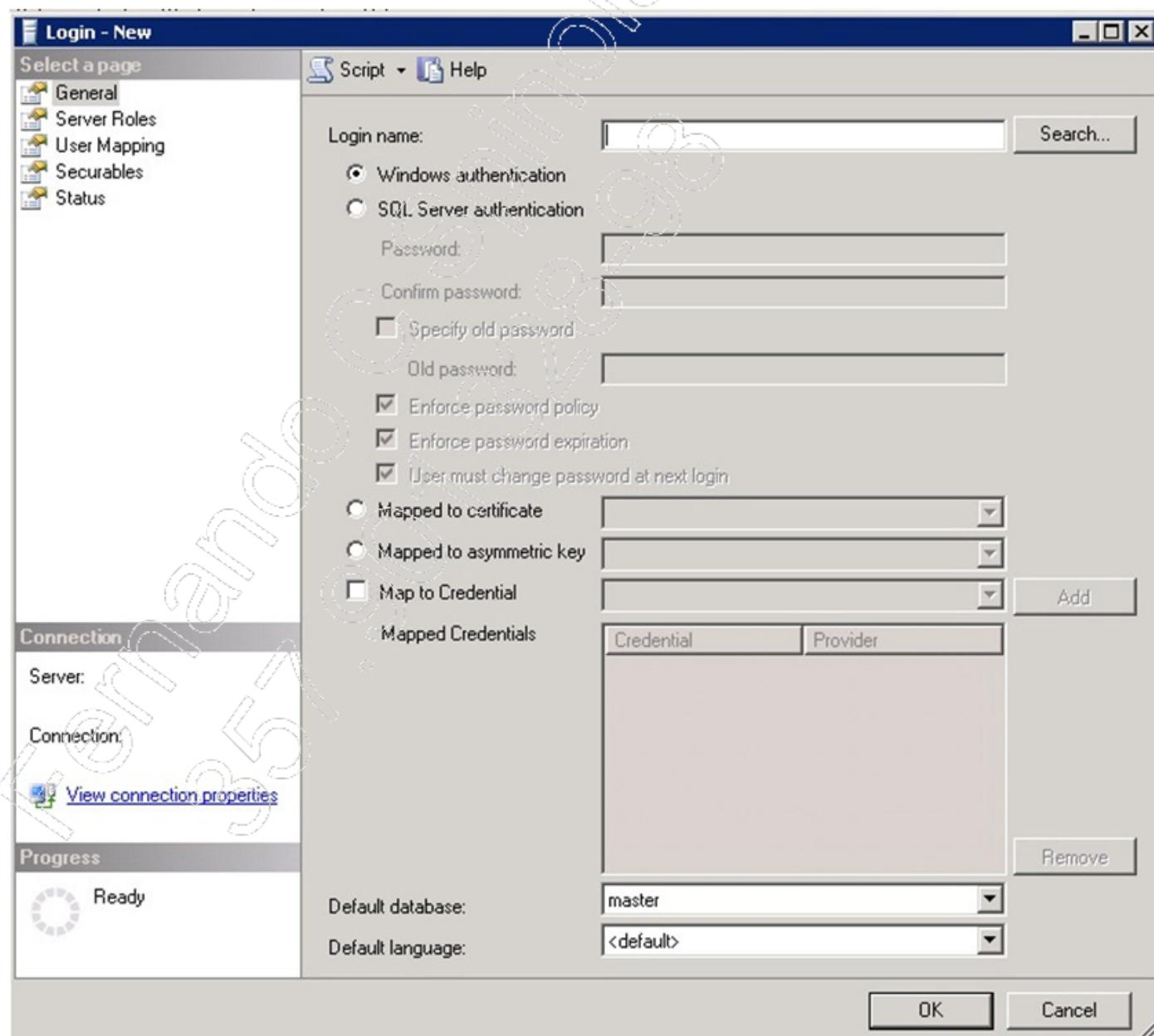
Podemos ainda criar logins no SQL Server associados a credenciais e a certificados.

- **Exemplo de criação de LOGIN no SSMS**

Selecione a opção logins, clique com o botão direito do mouse sobre ela e escolha New Login.



Selecione o tipo de autenticação do usuário: Windows authentication ou **SQL Server authentication**. Conforme o tipo escolhido, informe o nome do login (**Login name**). Caso seja **SQL SERVER**, defina uma senha (**Password**) e confirme-a (**Confirm password**). Ainda para usuários com autenticação pelo SQL Server, deve-se selecionar se deseja reforçar o uso da política de senhas (**Enforce password policy**), se a senha deve ser expirada de tempos em tempos (**Enforce password expiration**) e se o usuário deve trocar as senha no próximo login (**User must change password at next login**). Pode-se ainda definir o nome do database padrão (**Default database**) e o padrão de linguagem utilizado pelo login, que, se não for definido, é o padrão do banco de dados.



7.2.4. Criando usuários

Existem duas formas de criarmos usuários no SQL Server:

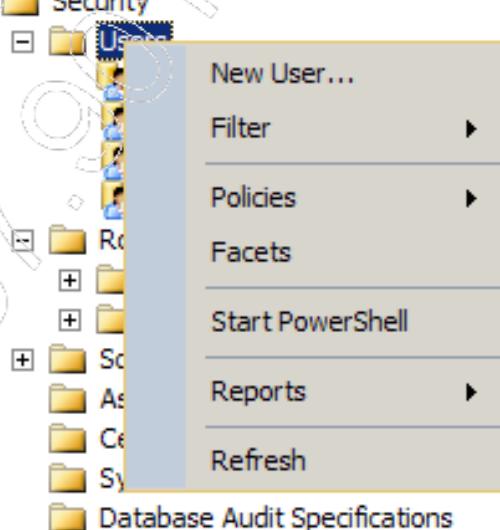
- **Através de comandos**

Para criar usuários, utilize a seguinte sintaxe:

```
CREATE USER user_name
  [
    { FOR | FROM } LOGIN login_name
  ]
  [ WITH DEFAULT_SCHEMA = schema_name ]
[ ; ]
```

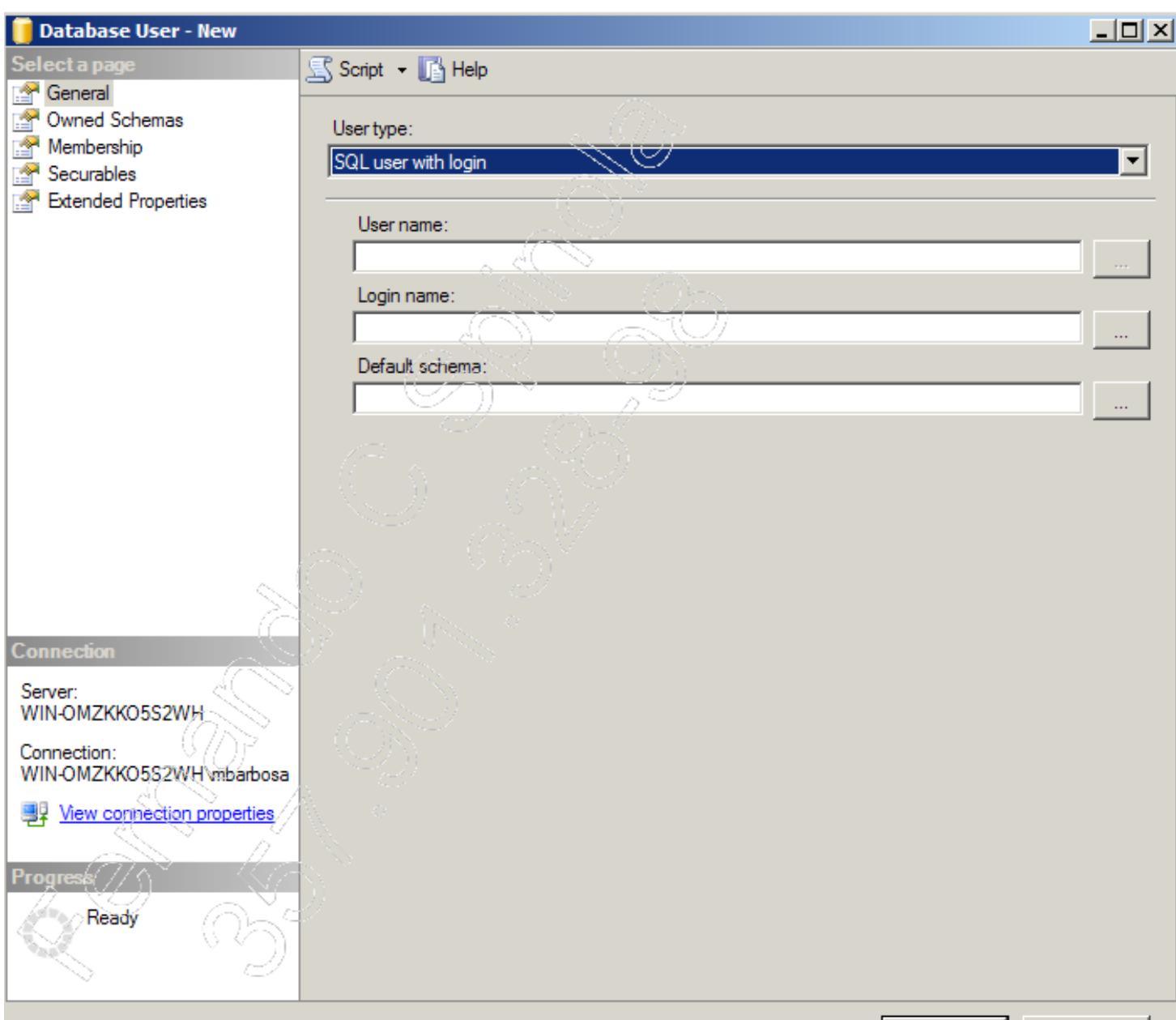
- **Pelo SSMS**

Selecione o banco de dados no qual deseja criar o usuário, clique com o botão direito do mouse sobre ele e escolha a opção **New User**.



SQL 2014 - Módulo III

Defina o tipo de usuário a ser criado (**User type**), indique o nome do usuário (**Username**) e o nome do login (**Login name**) ao qual o usuário estará relacionado. Recomenda-se definir o schema ao qual o usuário estará associado, pois, caso não seja informado, o usuário será associado ao schema chamado **dbo**.



7.2.5. Logins default

O SQL Server tem como login padrão o SA (System Administrator), que é o usuário com todos os poderes de administração do SQL Server. Esse padrão deve ser usado apenas para aplicação de patches e upgrades de versão, pois é vulnerável no que diz respeito à segurança. No caso de tentativa de invasão de um servidor SQL Server, por exemplo, este é um dos LOGINS que sofre tentativa de conexão. Recomenda-se renomear este LOGIN e deixá-lo desabilitado.

O segundo login default é o **BUILTIN\administrators**, que está associado aos administradores da máquina. Esse login requer certo cuidado, pois os administradores de um servidor Windows que tenha SQL Server poderão se conectar ao SQL Server como administradores da instância SQL Server. Recomenda-se desabilitar esse LOGIN e criar LOGINS para cada necessidade de acesso à instância SQL Server.

O login **GUEST** deve ser eliminado, pois permite que logins que não foram adicionados ao banco de dados accessem este banco através desta conta.

7.3. Gerenciando acesso à instância

Através das chamadas **FIXED SERVER ROLES**, pode-se atribuir maior grau de permissão de acesso. Como são dados no nível da instância, esses direitos são aplicáveis à instância e aos respectivos bancos de dados que a compõem.

Vejamos a lista das **FIXED SERVER ROLES**:

- **SYSADMIN**: Executa todas as atividades no SQL Server. É a permissão dada ao login SA. Possui todas as permissões das demais **FIXED SERVER ROLES**;
- **DBCREATOR**: Pode realizar a criação e alteração de bancos de dados da instância;
- **DISKADMIN**: Realiza o gerenciamento dos arquivos que compõem os bancos de dados da instância;
- **PROCESSADMIN**: Pode gerenciar os processos que estão em execução no SQL Server;
- **SECURITYADMIN**: Gerencia os logins na instância;
- **SERVERADMIN**: Gerencia as configurações da instância;
- **SETUPADMIN**: Gerencia as extended stored procedures executadas e suas permissões na instância e bancos de dados;
- **BULKINSERT**: Executa comandos Bulk Insert nos bancos de dados da instância.

Podemos adicionar uma FIXED SERVER ROLE a um login por meio do SSMS e por meio de comandos. Vamos conhecer alguns comandos a seguir:

- Para exibir a lista das server roles existentes:

```
EXEC SP_HELPSRVROLE
```

- Para exibir as permissões de cada FIXED SERVER ROLE existente:

```
EXEC SP_SRVROLEPERMISSION
```

- Para exibir os membros das FIXED SERVER ROLES:

```
EXEC SP_HELPSRVROLEMEMBER
```

- Para consultar as server roles atribuídas aos logins:

```
select p.name, p.type_desc, pp.name, pp.type_desc  
from sys.server_role_members roles  
join sys.server_principals p  
on roles.member_principal_id = p.principal_id  
join sys.server_principals pp  
on roles.role_principal_id = pp.principal_id
```

7.4. Gerenciando acesso aos bancos de dados

Para determinar as permissões de acesso a cada banco de dados, podemos definir, quando for o caso, o uso das **FIXED DATABASE ROLES**.

Vejamos a lista das FIXED DATABASE ROLES:

- **PUBLIC**: Mantém todas as permissões padrão do SQL Server, permite fazer a leitura de todas as visões do catálogo SQL Server;
- **DB_OWNER**: Executa qualquer atividade no banco de dados, menos DROP DATABASE;
- **DB_ACCESSADMIN**: Adiciona e remove usuários do banco de dados;
- **DB_DDLADMIN**: Adiciona, altera ou remove objetos do banco de dados;
- **DB_SECURITYADMIN**: Atribui permissões aos usuários;
- **DB_BACKUPOPERATOR**: Realiza operações de backup e restauração do banco de dados;
- **DB_DATAREADER**: Executa a leitura de qualquer tabela ou visão do banco de dados;
- **DB_DATAWRITER**: Executa INSERT, UPDATE, DELETE e MERGE de qualquer tabela do banco de dados;
- **DB_DENYDATAREADER**: Impede a leitura de qualquer tabela do banco de dados;
- **DB_DENYDATAWRITER**: Impede a execução de INSERT, UPDATE, DELETE e MERGE de qualquer tabela do banco de dados.

As permissões neste nível são dadas apenas aos usuários, não para os logins. Como existe a relação entre eles, o login acaba recebendo as permissões do usuário.

- Considerações a respeito das FIXED DATABASE ROLES:
 - Não podem ser removidas;
 - Qualquer membro de uma role pode adicionar outro membro;
 - A permissão da role será atribuída automaticamente ao executar o procedimento **SP_ADDROLEMEMBER**;
 - Podemos criar um novo grupo de permissões em um database (USER DEFINED DATABASE ROLES) e podemos atribuir a ele uma FIXED SERVER ROLE;
 - A atribuição de permissões a um grupo de permissões, e assim sucessivamente, pode provocar uma séria perda de performance;
 - As permissões estão disponíveis na visão **sysusers** de cada banco de dados.

Para atribuirmos direitos a usuários, precisamos criá-los. Podemos fazer isso por meio do SSMS e de comandos, como vimos anteriormente. Vamos conhecer mais alguns comandos a seguir:

- Para atribuirmos as permissões aos usuários:

```
EXEC SP_ADDROLEMEMBER ROLE, USER
```

- Para removermos as permissões aos usuários:

```
EXEC SP_DROPROLEMEMBER ROLE, USER
```

- Para consultarmos a lista de grupos de permissões existentes:

```
EXEC SP_HELPROLE
```

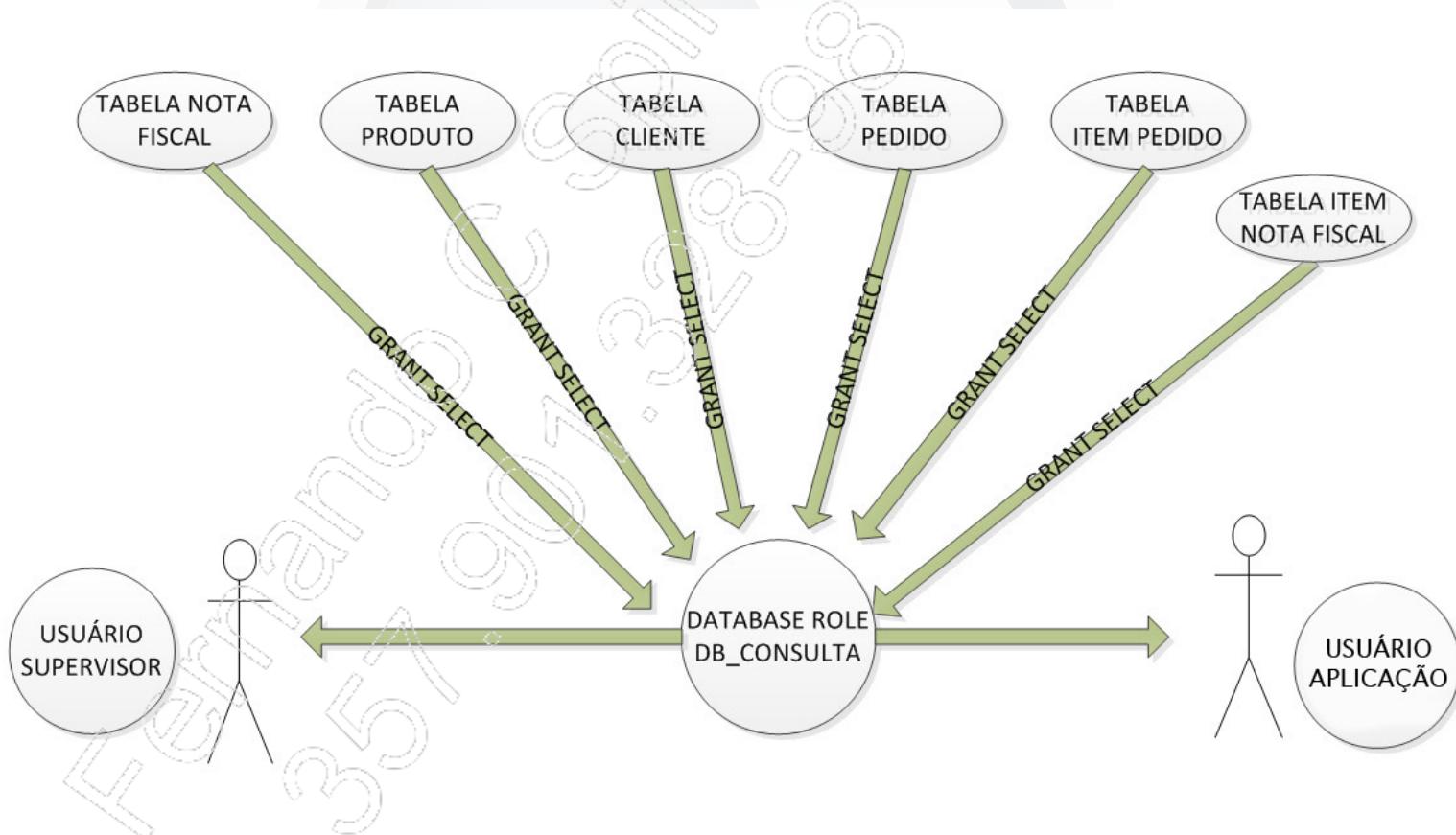
- Para consultarmos os membros de grupos de permissões existentes:

```
EXEC SP_HELPROLEMEMBER
```

7.5. Grupos de permissões criados pelo usuário

Conforme vimos, existem as permissões chamadas de FIXED SERVER ROLES e as DATABASE SERVER ROLES. A elas juntamos as permissões chamadas de USER DEFINED DATABASE ROLES, que são grupos de permissões criados pelo administrador para cada banco de dados para agrupar direitos que devem ser dados a principais.

A imagem a seguir demonstra a criação de um grupo de permissão criado pelo usuário, no qual é dado o direito de acesso a várias tabelas de consulta para a role chamada **DB_CONSULTA** e esta role é associada a dois usuários (**APLICACAO** E **SUPERVISOR**).



Essa operação em que se cria USER DATABASE ROLES é bastante importante para simplificar a atribuição de direitos e a consequente diminuição da quantidade de direitos a serem dados a usuários.

Vamos trabalhar um exemplo em que temos 500 tabelas e 100 usuários. O número individual de permissões seria $500 \times 100 = 500000$ permissões. Caso utilizássemos as USER DEFINED DATABASE ROLES, teríamos $500 + 100 = 600$ permissões.

Vejamos algumas sintaxes a seguir:

- Para criarmos este tipo de role:

```
Exec SP_ADDROLE nome_role, Owner  
Ou  
CREATE ROLE [schema.]nome_role
```

- Para adicionarmos um membro à role:

```
Exec SP_ADDROLEMEMBER nome_role, nome_usuario
```

- Para retirarmos um membro de uma role:

```
Exec SP_DROPROLEMEMBER nome_role, nome_usuario
```

- Para listarmos as Database Roles existentes:

```
Exec SP_HELPROLE
```

- Para listarmos as Database Roles existentes:

```
Exec SP_HELPROLE
```

SQL 2014 - Módulo III

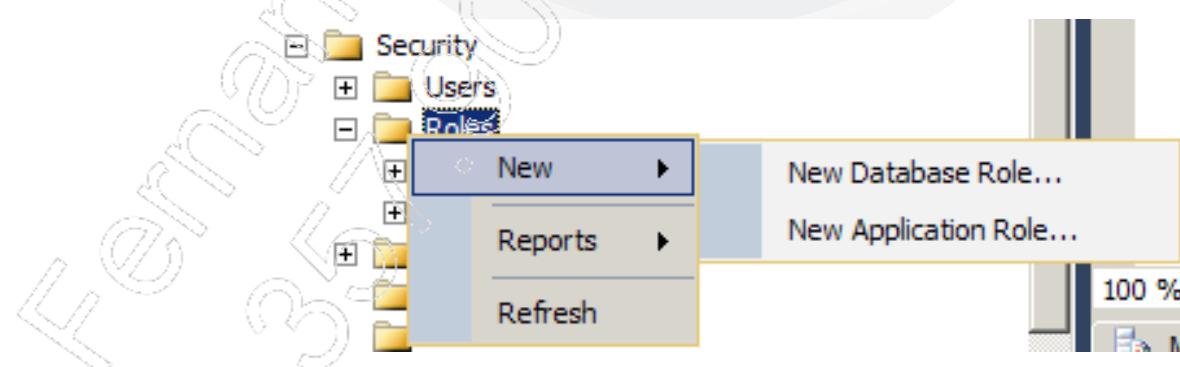
- Para eliminarmos uma Database Role existente:

```
Exec SP_DROPROLE nome_role  
Ou  
DROP ROLE [schema.]nome_role
```

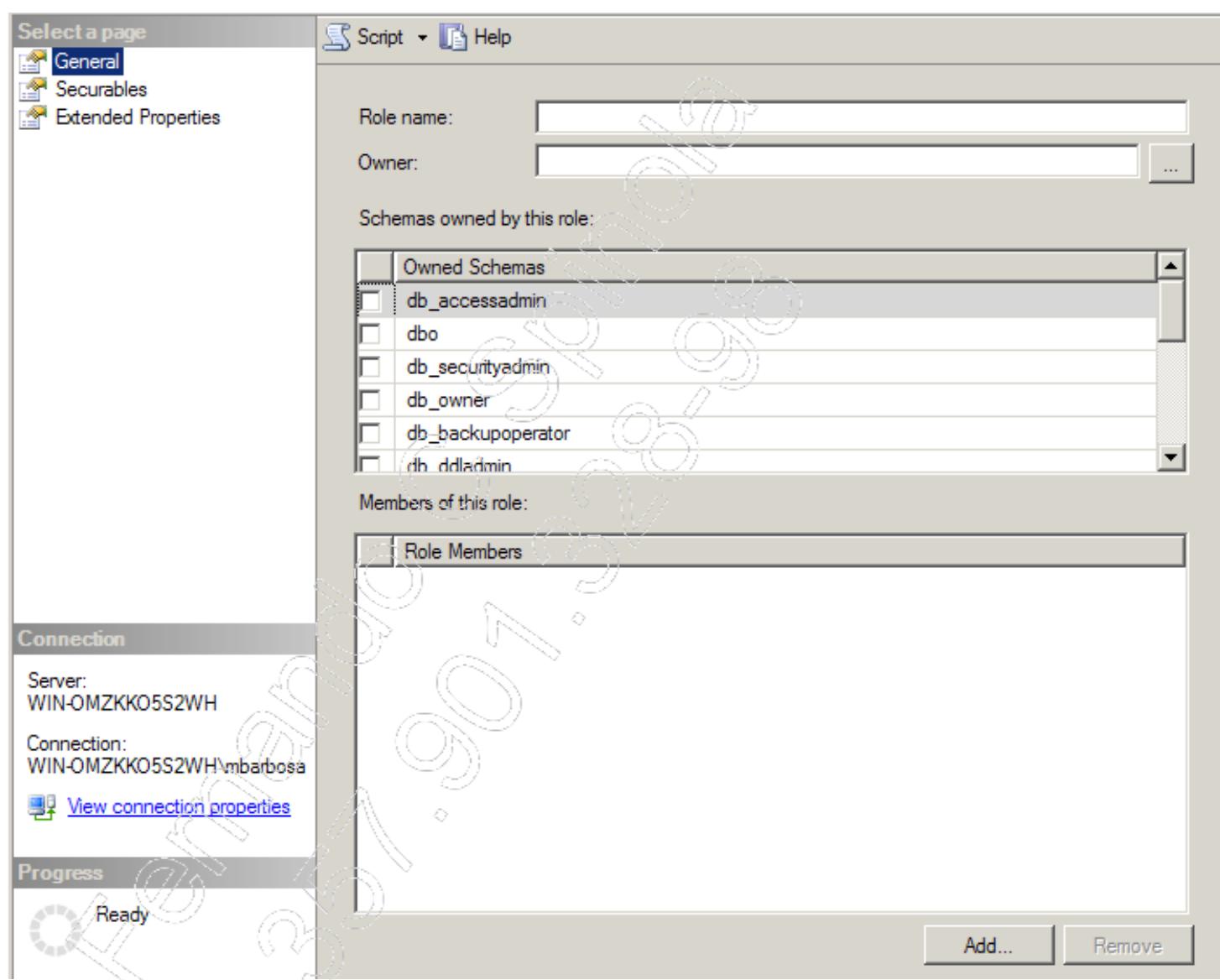
- Para consultarmos as Database Roles:

```
SELECT  
    p.name, p.type_desc, pp.name, pp.type_desc, pp.is_fixed_role  
FROM sys.database_role_members roles  
JOIN sys.database_principals p  
ON roles.member_principal_id = p.principal_id  
JOIN sys.database_principals pp  
ON roles.role_principal_id = pp.principal_id
```

Para criarmos uma Database Role usando SSMS, selecione o banco de dados desejado, escolha a pasta **security** e a opção **roles**. Em seguida, clique com o botão direito do mouse e selecione **New Database Role**:

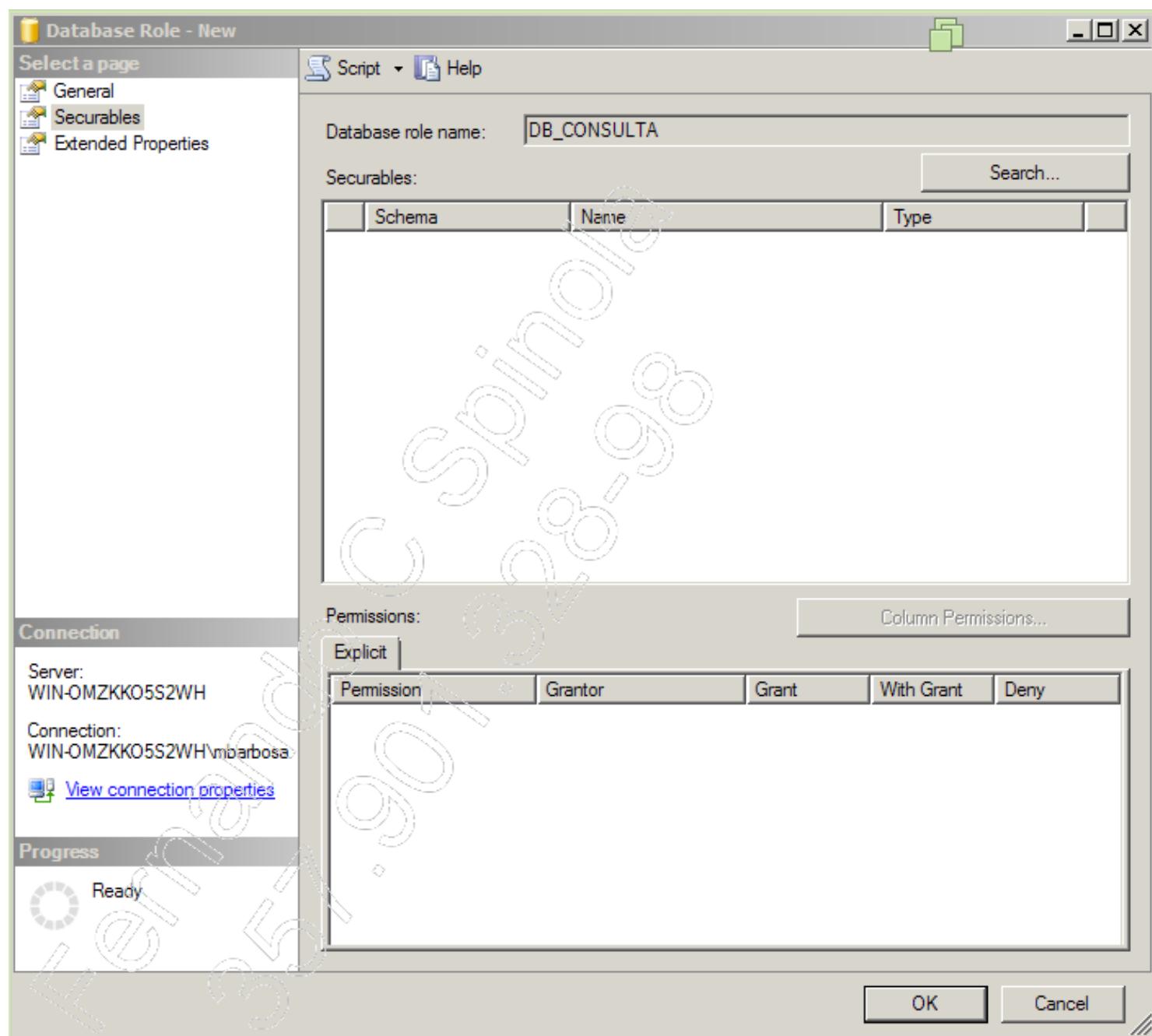


Indique o nome da role (**Role name**) e o dono da role (**Owner**), selecione os schemas que poderão utilizar a role (**Schemas owned by this role**) e clique no botão **Add...** para adicionar novos membros para a role.

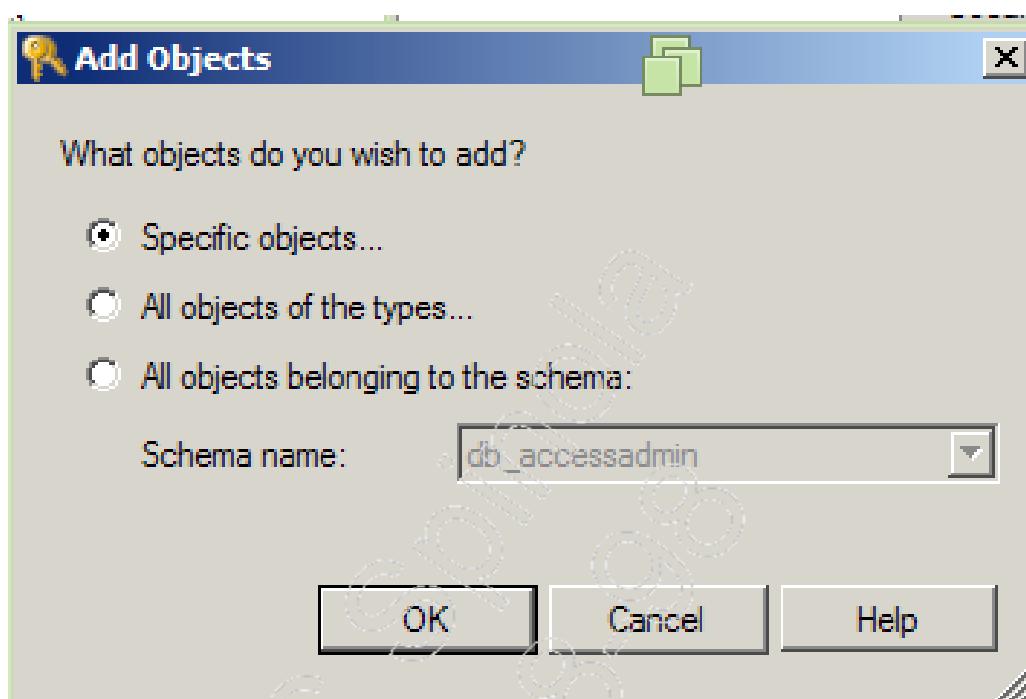


SQL 2014 - Módulo III

Selecione a opção **Securables** para adicionar os objetos e suas respectivas permissões:



Clique no botão **Search** e identifique os tipos de objetos que deseja associar à role. Selecione uma das opções: objetos específicos, todos os objetos de um tipo ou todos os objetos pertencentes a um schema.

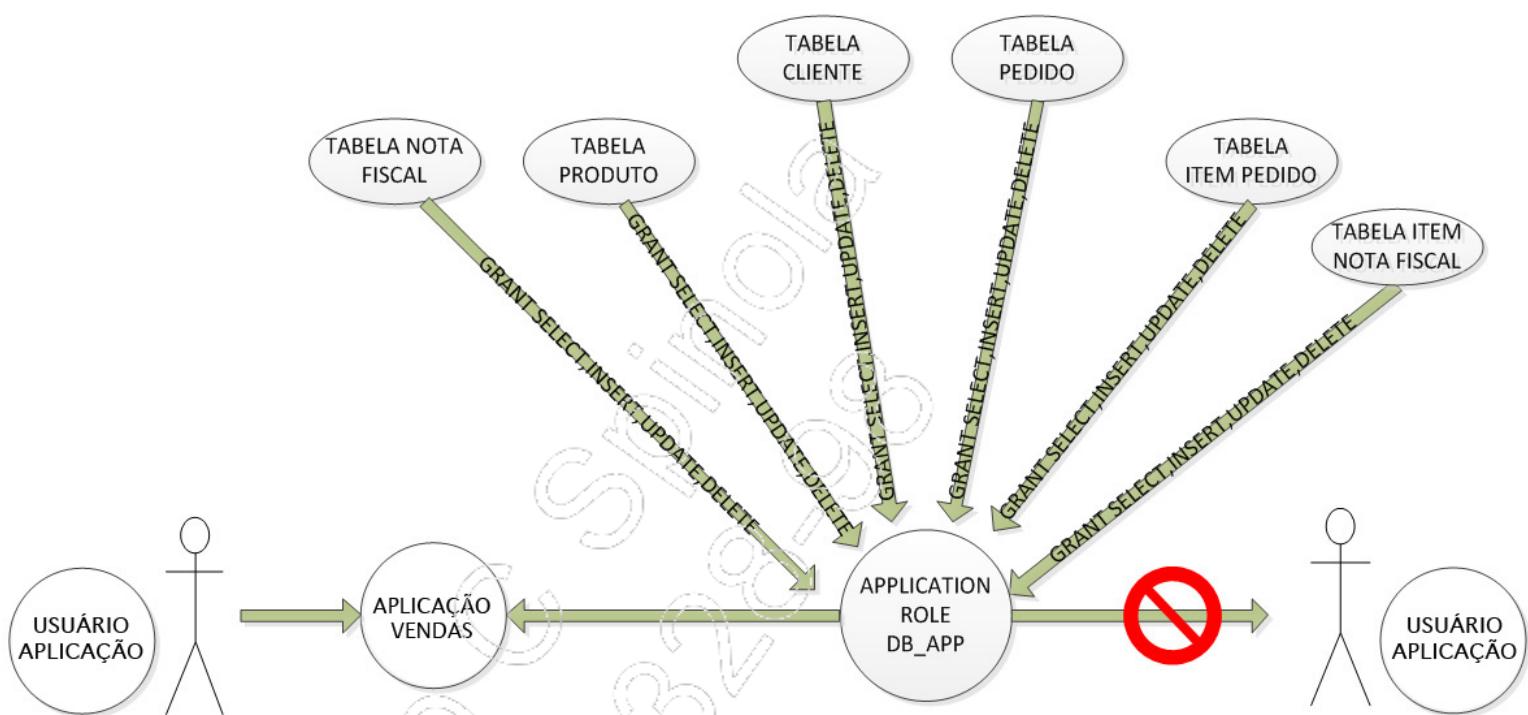


7.6. Grupos de permissões criados para aplicações

Quando precisamos impedir acessos indesejados através de ferramentas que não sejam somente a aplicação, podemos utilizar as **APPLICATION ROLES**, ou também **APP_ROLES**.

A aplicação e os objetos (tabelas, visões e outros) podem estar vinculados à APPLICATION ROLE. Esta, por sua vez, fica vinculada à aplicação, não a um usuário. Logo, se o usuário utilizar outra ferramenta, como Excel ou o próprio SSMS, não terá acesso aos objetos.

A figura a seguir ilustra que o usuário não consegue usufruir dos direitos da role diretamente. Para isso, ele deve utilizar a aplicação VENDAS que está, por sua vez, vinculada à role DB_APP.



Vejamos algumas sintaxes a seguir:

- Para criar a Application Role:

```
Exec SP_ADDAPPROLE nome_role, senha  
Ou  
CREATE APPLICATION ROLE [nome_role] WITH DEFAULT_SCHEMA = [nome_schema], PASSWORD = N'senha'
```

- Para eliminar uma Application Role:

```
Exec SP_DROPAPPROLE nome_role  
Ou  
DROP APPLICATION ROLE [nome_role]
```

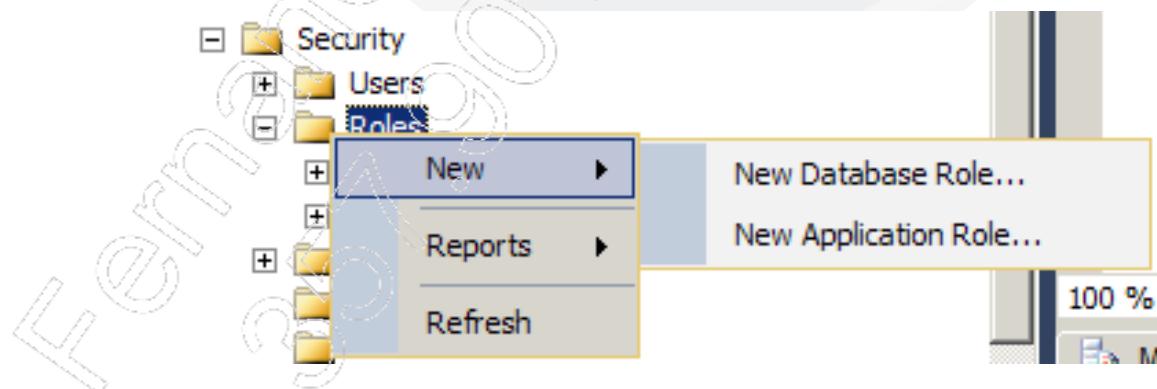
- Para ativar uma Application Role:

```
Exec SP_SETAPPROLE nome_role, senha
```

- Para listar as Application Roles existentes:

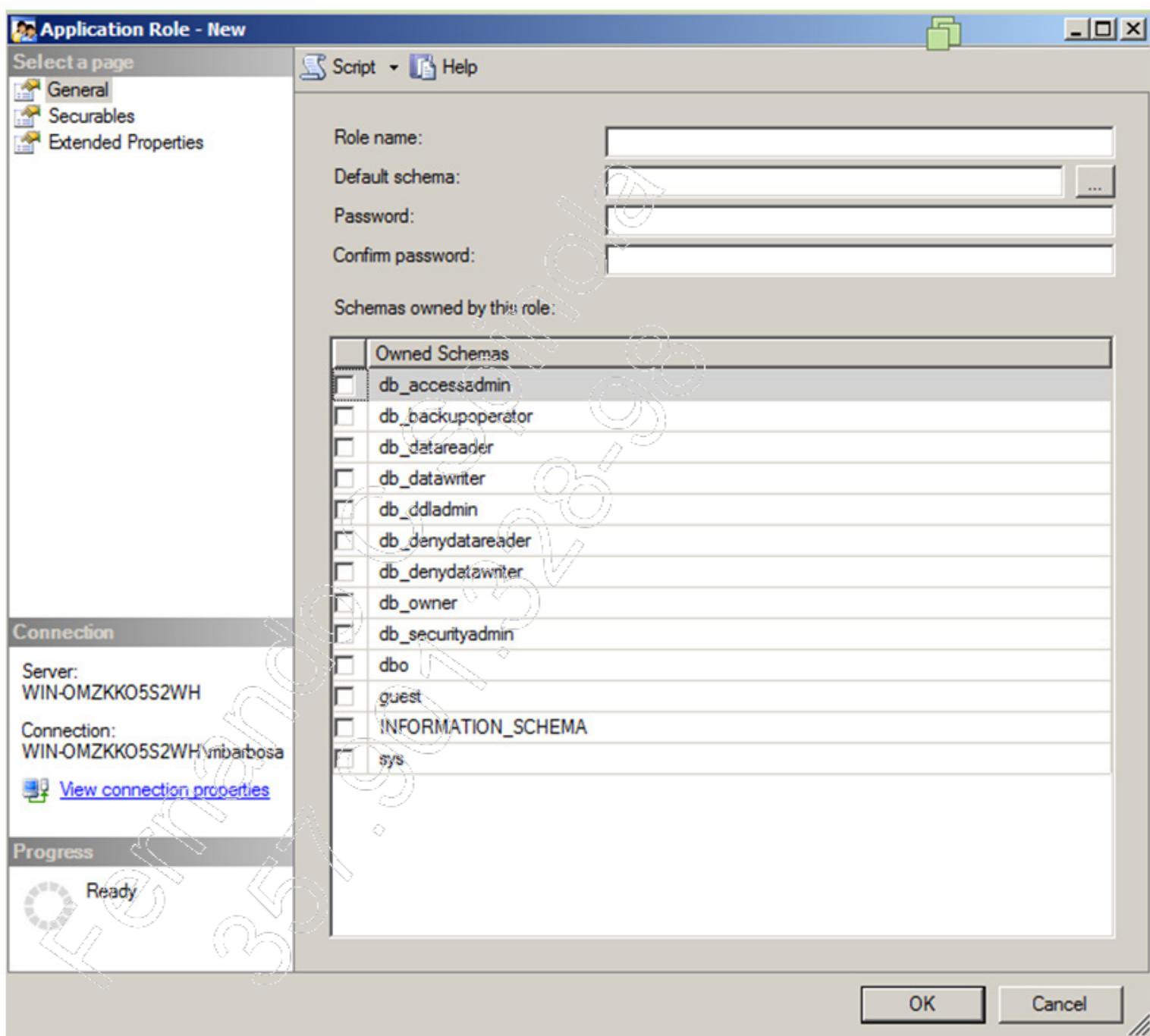
```
Exec SP_HELPROLE
```

Para criar uma Application Role usando SSMS, selecione o banco de dados desejado, escolha a pasta **Security** e a opção **Roles**. Em seguida, clique com o botão direito do mouse sobre essa opção e selecione **New / New Application Role**.



SQL 2014 - Módulo III

Indique o nome da role (**Role name**), o nome do schema default, informe a senha (**Password**) e confirme-a (**Confirm Password**). Em seguida, selecione os schemas que estão associados à Role (**Schemas owned by this role**).



7.7. Permissionamento

As permissões são tratadas, como vimos anteriormente, por três comandos:

- GRANT
- REVOKE
- DENY

As permissões estão divididas em três tipos:

- Permissões definidas para roles;
- Permissões de comandos;
- Permissões de objetos.

! Para que possa ser usada a cláusula WHERE em comandos UPDATE e DELETE, é necessário que seja dada a permissão de SELECT, além das permissões de UPDATE e DELETE.

Para listarmos todas as permissões atribuídas em um banco de dados SQL Server, podemos usar a consulta a seguir:

```
SELECT
    dp.class_desc, dp.permission_name, dp.state_desc,
    ObjectName = OBJECT_NAME(major_id),
    GranteeName = grantee.name,
    GrantorName = grantor.name
FROM sys.database_permissions dp
JOIN sys.database_principals grantee
    on dp.grantee_principal_id = grantee.principal_id
JOIN sys.database_principals grantor
    on dp.grantor_principal_id = grantor.principal_id
```

7.7.1.GRANT

O comando grant é o que utilizamos para atribuir direitos a logins e usuários.

A sintaxe para o comando **GRANT** é:

- Para permissão de comando:

```
GRANT {ALL | Permissao_comando, [...n] }  
TO LOGIN|ROLE|USUARIO|GRUPO USUARIO WINDOWS
```

- Para permissão de objeto:

```
GRANT {ALL [PRIVILEGES] | Permissao.[,...n] }  
[(column[,...n])] ON {TABELA|VISAO} |  
ON {stored procedure | extended procedure}  
TO LOGIN|ROLE|USUARIO|GRUPO USUARIO WINDOWS  
[WITH GRANT OPTION]
```

A cláusula **WITH GRANT OPTION** indica que o recebedor do direito poderá repassá-lo para outros, inclusive com a condição **WITH GRANT OPTION**.

7.7.2. DENY

Em função da herança e dos grupos de permissões, pode-se explicitamente negar um determinado privilégio. Isso indica que, apesar de ter sido concedido, o direito foi negado. Imaginemos que um usuário tenha acesso a uma tabela através de uma **USER DEFINED DATABASE ROLE**. Para impedir que ele a delete, por exemplo, podemos negar esse direito diretamente ao usuário. Isso fará com que ele tenha o direito, que fora concedido através do grupo de permissões, negado diretamente.

- Para negar a permissão de comando:

```
DENY {ALL | Permissao_comando, [...n] }  
TO LOGIN|ROLE|USUARIO|GRUPO USUARIO WINDOWS
```

- Para negar a permissão de objeto:

```
DENY {ALL [PRIVILEGES] | Permissao.[,...n] }  
[(column[,...n])] ON {TABELA|VISAO} |  
ON {stored procedure | extended procedure}  
TO LOGIN|ROLE|USUARIO|GRUPO USUARIO WINDOWS  
[CASCADE]
```

A cláusula **cascade** deve ser usada para negar permissão a um usuário que tenha recebido permissão **WITH GRANT OPTION**.

7.7.3. REVOKE

Podemos revogar uma permissão concedida com o comando **GRANT** ou revogar uma negação resultante do comando **DENY** utilizando o comando **REVOKE**. Em alguns aspectos, ele é semelhante ao comando **DENY**, porém, deve ficar claro que **REVOKE** é considerado como revogação completa, enquanto **DENY** é considerado uma revogação parcial.

O **GRANT** pode conceder um determinado privilégio e, em seguida, negá-lo através do comando **DENY**. Caso seja aplicado o **REVOKE**, esse comando afetará o **DENY** fazendo com que o privilégio seja retomado. Ao remover os direitos, eles são eliminados da visão **sys.sysprotects**. Somente podem revogar permissões os logins que forem pertencentes às FIXED SERVER ROLES (**sysadmin**, **db_owner** e **db_securityadmin**) e os proprietários de objetos. Isso vale para os comandos **GRANT**, **DENY** e **REVOKE**.

- Para revogar a permissão de comando:

```
REVOKE {ALL | Permissao_comando, [...] }  
TO LOGIN|ROLE|USUARIO|GRUPO USUARIO WINDOWS
```

- Para revogar a permissão de objeto:

```
REVOKE {ALL [PRIVILEGES] | Permissao.[,...n] }  
[(column[,...n])] ON {TABELA|VISAO} |  
ON {stored procedure | extended procedure}  
TO LOGIN|ROLE|USUARIO|GRUPO USUARIO WINDOWS  
[CASCADE]
```

7.8. Schema

A maioria dos objetos criados no SQL Server pertence a alguém (proprietários). Ao conjunto de objetos que um usuário possui é dado o nome de **SCHEMA**.

Usando o exemplo “João tem um celular e um MP3 Player”, podemos entender que João seria um usuário e o celular pertenceria a ele, assim como o MP3 Player. Logo, os dois equipamentos representam, de forma análoga, o Schema do usuário João.

Os objetos devem ser tratados primeiramente pelo seu proprietário (**criador**) e depois pelo nome do objeto, por exemplo, **produção.cliente**.

Existem alguns benefícios da relação usuário-schema:

- Vários usuários podem estar associados a um mesmo schema, facilitando a atribuição de direitos a todos os usuários que compartilham o referido schema;
- Ao eliminarmos um usuário, ele não possui objetos, mas sim o schema;
- Vários usuários podem compartilhar objetos sem a necessidade de identificar schemas diferentes;
- Facilidade na manutenção dos privilégios para grandes grupos de usuários;
- O nome totalmente qualificado de um objeto é dado da seguinte forma: **server.database.schema.objeto**;
- A separação dos conceitos de schema e usuário facilita a atribuição de direitos aos usuários. Se desejarmos eliminar um usuário, ele não possuirá objetos, logo a exclusão será possível.

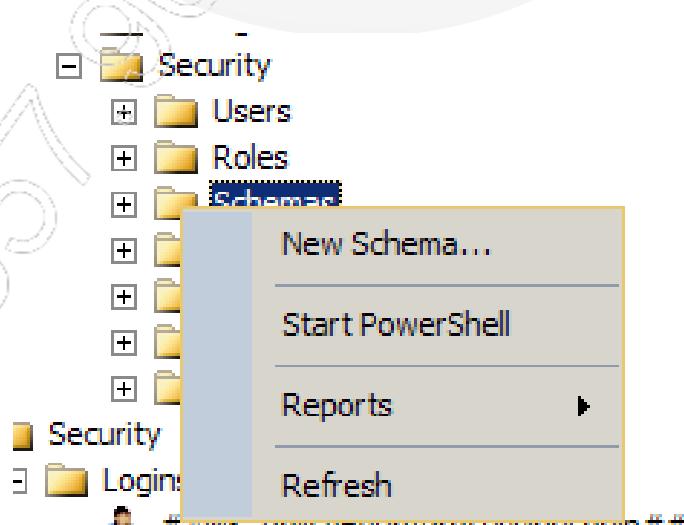
SQL 2014 - Módulo III

Existe no banco de dados SQL Server um schema padrão criado chamado de DBO. Caso um usuário crie um objeto e não especifique seu schema, este objeto fará parte do schema DBO.

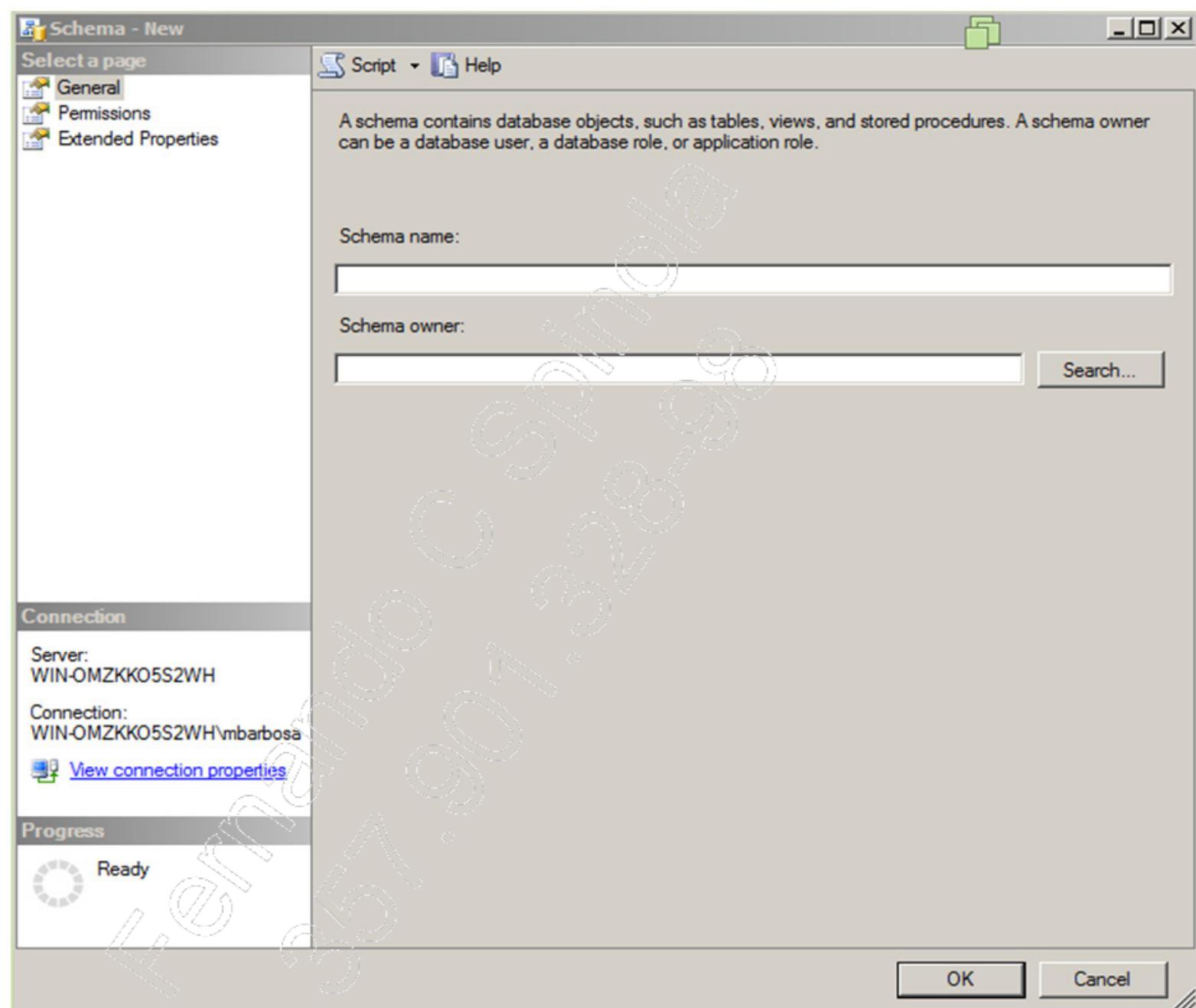
Vejamos a sintaxe do comando **CREATE SCHEMA**:

```
CREATE SCHEMA schema_name_clause [ <schema_element> [ ...n ] ]  
  
<schema_name_clause> ::=  
{  
    schema_name  
    | AUTHORIZATION owner_name  
    | schema_name AUTHORIZATION owner_name  
}  
  
<schema_element> ::=  
{  
    table_definition | view_definition | grant_statement |  
    revoke_statement | deny_statement  
}
```

Podemos visualizar os schemas utilizando a visão **sys.schemas**. Para criar um schema usando o SSMS, selecione o banco de dados desejado, expanda a pasta **Security**, clique com o botão direito do mouse sobre a opção **Schemas** e selecione **New Schema**.



Defina o nome, em **Schema Name**, e o proprietário do schema, em **Schema owner**.



7.9. Credenciais

Esse recurso permite que o SQL Server acesse recursos externos, por exemplo, usando programação .NET. Por padrão, logins do SQL Server não podem acessar nenhum tipo de recurso externo. Podemos atribuir as credenciais de um usuário Windows para um determinado login.

Vejamos a sintaxe para criação de uma credencial:

```
CREATE CREDENTIAL nome_credencial  
WITH IDENTITY = 'nome_da_identidade'  
SECRET = 'segredo'
```

Em que:

- **Nome_credencial**: Especifica o nome da credencial;
- **IDENTITY = 'nome_da_identidade'**: Nome da conta a ser utilizada ao conectar a um recurso externo do SQL Server;
- **SECRET**: Especifica o segredo para autenticação externa. Sendo esta cláusula opcional, caso o identity seja um usuário Windows, **secret** será a sua senha.

Para alterar uma credencial:

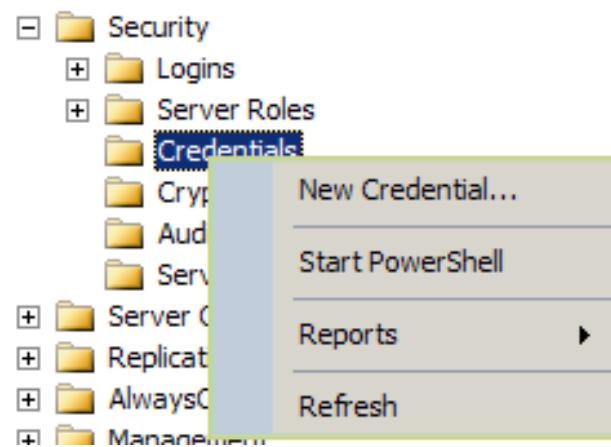
```
ALTER CREDENTIAL nome_credencial  
WITH IDENTITY = 'nome_da_identidade'  
SECRET = 'segredo'
```

Para eliminar uma credencial:

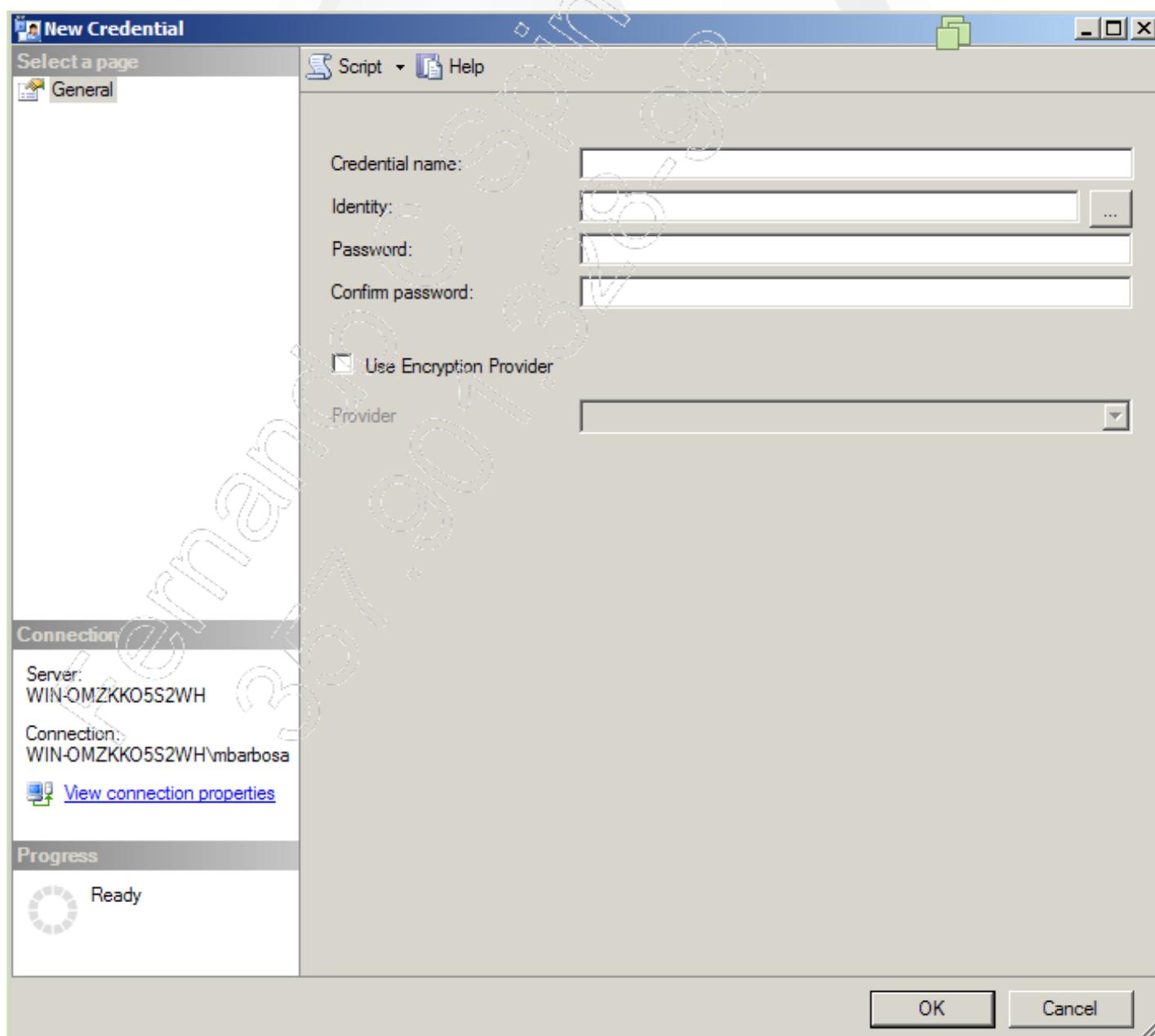
```
DROP CREDENTIAL nome_credencial
```

Para visualizar informações sobre credenciais, você pode utilizar a visão **sys.credentials**.

Para criar credenciais usando o SSMS, expanda a pasta **Security** da instância e selecione a opção **Credentials / New Credential**.



Informe o nome da credencial em **Credential name**, defina a identidade em **Identity**, indique a senha em **Password** e confirme-a em **Confirm password**.



Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo.

- Para acessar um banco de dados determinado, é necessário que um login tenha mapeado um usuário específico para o banco de dados;
- Permissões podem ser dadas individualmente a um usuário ou role ou através de roles, sendo de caráter cumulativo;
- O comando **DENY** nega um direito dado e o comando **REVOKE** revoga um comando **GRANT** ou **DENY** (o último que for aplicado);
- Não se pode revogar um direito diferente do que foi dado a um usuário;
- Esquema (schema) simplifica a criação e a manutenção de objetos no SQL Server;
- O esquema DBO existe em todas as versões do SQL Server;
- Credenciais tornam possível o acesso a recursos externos por parte do SQL Server;
- O usuário que criar um objeto tem, por padrão, todas as permissões sobre esse objeto.

7

Segurança de dados

Teste seus conhecimentos

Fernando Spinola
357.907.2698



IMPACTA
EDITORA

1. Que recurso do SQL Server permite o acesso a recursos externos?

- a) Polices
- b) LOGIN
- c) USER
- d) Credenciais
- e) Principals

2. O que acontece quando executamos o comando REVOKE após um comando DENY tratando dos mesmos direitos?

- a) Não ocorre nenhuma ação.
- b) O comando REVOKE revogará o comando DENY.
- c) Gera um erro de sintaxe.
- d) Nega acesso ao recurso.
- e) Não funciona com DENY.

3. Quais são os tipos de Roles existentes?

- a) FIXED SERVER e APPLICATION.
- b) FIXED SERVER, DATABASE e USER DATABASE.
- c) FIXED SERVER, APPLICATION e DATABASE.
- d) APPLICATION, DATABASE e USER DATABASE.
- e) FIXED SERVER, APPLICATION, DATABASE e USER DATABASE.

4. Qual afirmação está errada sobre schema?

- a) É o proprietário do objeto.
- b) Vários usuários podem estar associados a um mesmo schema.
- c) Não é possível eliminar um usuário que possui schema.
- d) Facilita a manutenção de privilégios.
- e) Ao eliminarmos um usuário, ele não possui objetos, mas sim o schema.

5. Por que é válida a separação dos conceitos de usuário e de schema?

- a) Não existe diferença.
- b) Ao eliminarmos um usuário, ele não possuirá objetos, logo a exclusão é possível.
- c) Não é possível separar usuário de schema.
- d) Não é válida e é recomendada a utilização somente de usuário.
- e) O objeto está associado ao usuário e não ao schema.

7

Segurança de dados

Mãos à obra!

Fernando Cipolla
357.907.328-98



IMPACTA
EDITORA

Observações iniciais

1 - O que será feito neste laboratório

Através de comandos:

- Criação de logins do Windows no SQL Server;
- Criação de logins do SQL Server;
- Criação de schemas;
- Atribuição de logins aos Fixed Server Roles;
- Criação de usuários com schemas default;
- Atribuição de usuários aos User Defined Roles;
- Conexão utilizando os usuários e criação das tabelas do banco Impacta;
- Atribuição de permissões aos usuários finais;
- Exibição de metadados;
- Eliminação de usuários;
- Eliminação de logins.

Através de ferramenta gráfica:

- Criação de logins do Windows no SQL Server;
- Criação de logins do SQL Server;
- Criação de schemas;
- Atribuição de logins aos Fixed Server Roles;
- Criação de usuários com schemas default;

- Atribuição de usuários aos User Defined Roles;
- Conexão utilizando os usuários e criação das tabelas do banco Impacta;
- Atribuição de permissões aos usuários finais.

2 – Scripts utilizados neste laboratório

A pasta **Capítulo_07** contém todos os scripts com os comandos necessários para a realização do laboratório 1. Utilize os scripts se não desejar digitar os respectivos comandos ou apenas para corrigir sua execução.

3 – Forma de conexão no sistema operacional

Para realizar os laboratórios deste capítulo, conecte-se ao Windows com a conta de aluno cuja senha é **password**. Para isso, utilize CTRL + ALT + DEL e escolha a opção Logoff. Em seguida, utilize CTRL + ALT + DEL novamente e conecte-se ao Windows com o login **Alunox** (em que x representa seu número de aluno em sala de aula).

Laboratório 1

A - Executando a sequência com comandos

1. Abra a ferramenta **Microsoft SQL Server Management Studio** e conecte-se ao sistema com **Windows Authentication**;
2. Na barra de ferramentas, clique no botão **New Query**;
3. Obtenha o nome do usuário corrente e verifique que ele deve ser o **dbo**;
4. Sabendo que os usuários e os grupos de usuários já estão previamente criados no Windows da máquina do instrutor (controlador de domínio), observe o quadro a seguir e execute as respectivas solicitações. A sequência em que estes passos devem ser executados é a seguinte:
 - 4.1. Crie no SQL Server os logins do Windows;
 - 4.2. Crie no SQL Server os logins internos, atribuindo a eles o password **xptoNomeUsuário**. Por exemplo, o password do usuário **Pedro** deverá ser **xptoPedro**;

Login Windows	Login SQL	Fixed Database Roles	User do Database
Jorge		Sysadmin	
Jairo			IMPACTA
Roberta			IMPACTA
Helena			IMPACTA
Daniel			IMPACTA
Fernando			IMPACTA
	Paula		IMPACTA
	Sonia		IMPACTA
	Raquel		IMPACTA

Login Windows	Login SQL	Fixed Database Roles	User do Database
Grupo VendedoresW • Ana • Lucas • Helder			IMPACTA
	Marcos	Sysadmin	
	Pedro		IMPACTA
	Carla		IMPACTA
	Margarida		IMPACTA
	Cláudio		IMPACTA

5. Observe o quadro anterior e realize as suas solicitações. A sequência adequada para executar este exercício é:

- 5.1. Conectado no SQL Server com a conta de aluno, acesse o database **Impacta**;
- 5.2. Crie os schemas **RH**, **VENDAS**, **FATURAMENTO** e **INSTRUTORES**;
- 5.3. Crie os usuários com seus **Schemas Default**, como mostra o quadro adiante;
- 5.4. Coloque os respectivos usuários nos **Fixed Database Roles**, como mostra o quadro adiante;
- 5.5. Crie o **User Defined Role** chamado **VendedoresSQL**;
- 5.6. Coloque dentro desse **User Defined Role** o grupo de usuários do Windows chamado **VendedoresW**;
- 5.7. Coloque o **User Defined Role** chamado **VendedoresSQL** nos **Fixed Database Roles db_datareader** e **db_datawriter**;

SQL 2014 - Módulo III

5.8. Coloque os devidos usuários nos seus respectivos **Fixed Database Roles** e no **User Defined Role VendedoresSQL**, como mostra o quadro a seguir:

User	Default Schema	Fixed DatabaseRole	User Def. Role
Dominio\Jairo	dbo	DB_Owner	
Dominio\Roberta	dbo	DB_DataReader DB_DataWriter	
Dominio\Helena	rh	DB_DDLAdmin	
Dominio\Daniel	vendas	DB_DDLAdmin	
Dominio\Fernando	faturamento	DB_DDLAdmin	
Dominio\VendedoresW			VendedoresSQL - db_datareader - db_datawriter
Pedro	dbo		VendedoresSQL - db_datareader - db_datawriter
Carla	dbo		VendedoresSQL - db_datareader - db_datawriter
Margarida	dbo		VendedoresSQL - db_datareader - db_datawriter
Cláudio	instrutores		
Paula	dbo		
Sonia	dbo		
Raquel	dbo		

6. Conecte-se ao Windows como **Helena** e crie as tabelas **Funcionario**, **Atributo** e **FuncAtrib**, como mostram os comandos a seguir:

```
-- conexão para Dominio\Helena

Use Impacta
go
SELECT USER
go
CREATE TABLE Funcionario
(
    Cod_Func int,
    Nome_Func      varchar(100)
)
go
CREATE TABLE Atributo
(
    Cod_Atrib     int ,
    Nome_Atrib  varchar(100)
)
go
CREATE TABLE FuncAtrib
(
    Cod_Func int,
    Cod_Atrib  int
)
go
```

SQL 2014 - Módulo III

7. Conecte-se ao Windows com a sua conta de aluno e, no SQL Server, faça a conexão como **Cláudio**, cujo password é **xptoClaudio**. Crie as tabelas **Instrutor**, **Treinamento** e **InstTrein**, como mostram os comandos a seguir:

```
--Conexão do Cláudio (SQL)

Use Impacta
go
SELECT user
go
CREATE TABLE Instrutor
(
    Cod_Instr      int,
    Nome_Instr    varchar(100)
)
go
CREATE TABLE Treinamento
(
    Cod_Trein      int,
    Nome_Trein    varchar(100)
)
go
CREATE TABLE InstTrein
(
    Cod_Trein      int,
    Nome_Trein    varchar(100)
)
go
```

8. Conecte-se no Windows como **Daniel** e crie as tabelas **Cliente** e **CliTrein**, como mostram os comandos a seguir:

```
--Conexão para Dominio\Daniel  
  
Use Impacta  
go  
SELECT USER  
go  
CREATE TABLE Cliente  
(  
    Cod_Cli int,  
    Nome_Cli varchar(100)  
)  
go  
CREATE TABLE CliTrein  
(  
    Cod_Cli int,  
    Cod_Trein int  
)  
go
```

SQL 2014 - Módulo III

9. Conecte-se ao Windows como **Fernando** e crie as tabelas **Fatura** e **NF**, como mostram os comandos a seguir:

```
--Conexão para Dominio\Fernando

Use Impacta
go
SELECT USER
go

CREATE TABLE NF
(
    Num_NF  int ,
    Val_NF   decimal(10,2),
    Data_NF  smalldatetime
)
go

CREATE TABLE Fatura
(
    Num_Fat int,
    Cod_Cli int,
    Num_NF   int,
    Data_Fat smalldatetime,
    Val_Fat   decimal(10,2)
)
go
```

10. Conecte-se ao SQL Server com autenticação do Windows. Acesse o database **Impacta** e execute o seguinte comando:

```
SELECT Sys.schemas.Name AS Nome_Schema,
       Sys.Tables.Name AS Nome_Tabela
  FROM Sys.Schemas inner join sys.Tables
    ON Sys.Schemas.schema_Id = Tables.Schema_Id
```

11. Note que as tabelas ficam associadas aos seus **schemas** e não ao usuário que as criou. Observe o quadro a seguir e atribua as permissões de **SELECT (S)**, **INSERT (I)**, **UPDATE (U)** e **DELETE (D)** nas respectivas tabelas;

Usuário	Permissões DML em tabelas			
	S	I	U	D
Paula	RH.Funcionario	X		
	RH.Atributo	X	X	
	RH.FuncAtrib	X	X	X
VendedoresSQL	VENDAS.Cliente	X	X	X
	INSTRUTORES.Treinamento	X		
	VENDAS.Clitrein	X	X	X

- VendedoresW
 - Pedro
 - Carla
- Margarida

SQL 2014 - Módulo III

Usuário		Permissões DML em tabelas			
		S	I	U	D
Sonia	VENDAS.Clitrein	X			
	FATURAMENTO.NF	X	X		
	FATURAMENTO.Fatura	X	X		
Raquel	INSTRUTORES.Instrutor	X	X		
	INSTRUTORES.InstTrein	X	X	X	
	INSTRUTORES.Treinamento	X			

12. Execute as procedures a seguir no database **Impacta** para obter os metadados destas configurações:

```
Exec SP_HelpUser  
Exec SP_HelpProtect  
Exec SP_HelpLogins
```

Laboratório 2

A - Executando a sequência de forma gráfica



Para executar este exercício, utilize os mesmos logins, usuários e permissões do laboratório anterior.

1. Abra a ferramenta **Microsoft SQL Server Management Studio** e conecte-se ao sistema com **Windows Authentication**;
2. Na barra de ferramentas, clique no botão **New Query**;
3. Obtenha o nome do usuário corrente e veja que ele deve ser o **dbo**;
4. Acesse o database **Impacta** e elimine todos os usuários criados neste database;
5. Acesse o database **Master** e elimine todos os **Logins** criados anteriormente;
6. Utilizando a ferramenta **Microsoft SQL Server Management Studio**, do lado direito da tela no **Object Explorer**, expanda a pasta **Security** (situada logo abaixo da pasta **Database**), clique com o botão direito sobre **Logins**, escolha a opção **New Login** e crie os logins do SQL Server e/ou do Windows. Para os logins do Windows, deixe marcada a opção **Windows Authentication**. Se for um login do SQL Server, marque a opção **SQL Server Authentication**. Para os usuários do SQL Server, é necessário colocar um password, que, neste caso, deverá ser como no **Laboratório 1**, ou seja, **xpto** seguido do próprio nome do usuário (ex.: **xptoPedro**). Para um usuário do SQL Server, desmarque a opção **User must change password at next login**. Para encerrar, clique no botão **OK**;
7. Repita a operação até criar todos os logins necessários;
8. Feche a pasta **Logins**;
9. Abaixo da pasta **Logins**, expanda a pasta **Server Roles**;

10. Clique com o botão direito do mouse sobre o **Role** em que deseja colocar o **Login** e escolha a opção **Properties**;
11. Na tela que será exibida, clique no botão **Add** para acrescentar o usuário dentro do respectivo role;
12. Clique no botão **Browse** da próxima tela para obter os **Logins** desejados para o **Role** em questão e, em seguida, clique em **OK**;
13. Repita a operação para todos os roles em questão;
14. Feche a pasta **Server Roles**;
15. Feche a pasta **Security**;
16. Expanda a pasta **Databases**;
17. Expanda o database **Impacta**;
18. Expanda a pasta **Security** que fica dentro do database **Impacta**;
19. Clique com o botão direito do mouse sobre **Users** e escolha a opção **New User**;
20. Na tela que será exibida, no campo **User Name**, escreva o nome do usuário e, no campo **Login Name**, clique nas reticências (...);
21. Clique no botão **Browse** da próxima tela e escolha, na lista exibida, o referido login, que será um usuário desse database. Escolha um login de cada vez e, em seguida, clique no botão **OK**;

22. No campo **Default Schema**, clique nas reticências (...);
23. Na tela que será exibida, clique no botão **Browse**;
24. Na lista que será exibida, escolha um **Schema default** para esse usuário e clique no botão **OK**;
25. Clique em **OK** novamente;
26. Na parte inferior desta tela, escolha os **Roles** aos quais esse usuário deverá pertencer;
27. Clique no botão **OK**;
28. Repita essa operação para cada um dos usuários que devem pertencer a esse database, segundo as indicações já apresentadas no **Laboratório 1**;
29. Assim que terminar, feche a pasta **Users**, a pasta **Security**, o database **Impacta** e a pasta **Databases**.

Automação de tarefas, alertas e operadores

8

- ✓ Conta do SQL Server Agent;
- ✓ Configurando o envio de e-mails através do SQL Server;
- ✓ Configurando tarefas (Jobs);
- ✓ Configurando operadores (Operators);
- ✓ Configurando alertas (Alerts);
- ✓ Configurando múltiplos SQL Server Agents utilizando centralização;
- ✓ Fazendo cópia de todas as tarefas;
- ✓ Solução de problemas (Troubleshooting).

8.1. Introdução

Muitas vezes, na administração de um banco de dados, o administrador responsável precisará que determinadas atividades sejam executadas periodicamente, como verificação de integridade dos arquivos de dados, criação de objetos, recriação de índices, backups, entre outras atividades. Muitas dessas atividades precisam ser realizadas fora do horário de trabalho, quando não há pessoal disponível para realizá-las. O SQL Server dispõe de recursos para atender a essas necessidades e disponibiliza o serviço chamado SQL Server Agent. Ele tem por finalidade automatizar a execução de tarefas, sendo elas agendadas ou não. Além disso, esse serviço proporciona um maior controle sobre as ocorrências de situações indesejadas (alerts), tais como:

- Uso excessivo de memória;
- Problemas de performance de I/O;
- Problemas de contenção de Transaction Log;
- Excesso de I/O em disco;
- Problemas de contenção de bloqueios;
- Problemas ligados a hot files;
- Problemas de I/O em áreas temporárias;
- Problemas ligados à replicação de dados;
- Falhas em nós do cluster;
- Falta de índices no acesso a dados;
- Entre outros.

Para auxiliar na execução de tarefas, o administrador pode configurar e-mails para envio de mensagens relativas às atividades agendadas (Jobs) ou para informação de ocorrências indesejadas. Além disso, é possível configurar o recurso de operadores, por meio dos quais são enviadas mensagens on-line para terminais de operação, tornando a visualização das mensagens mais eficientes.

Os principais requisitos para a execução do SQL Server Agent é a configuração dos serviços no Windows utilizando uma conta adequada (com um usuário gerenciado pelo AD – Active Directory). Normalmente, a inicialização do serviço do SQL Agent deve ocorrer após a inicialização do banco de dados. Recomenda-se que este serviço tenha inicialização automática, pois, em caso de queda inesperada do servidor, o serviço é automaticamente retomado.

Existe a possibilidade de que, em ambientes que disponham de múltiplas instâncias SQL Server, haja a centralização do SQL Server Agent, de tal forma que o administrador possa realizar um controle central de todos os SQL Agents de todas as instâncias gerenciadas.

8.2. Conta do SQL Server Agent

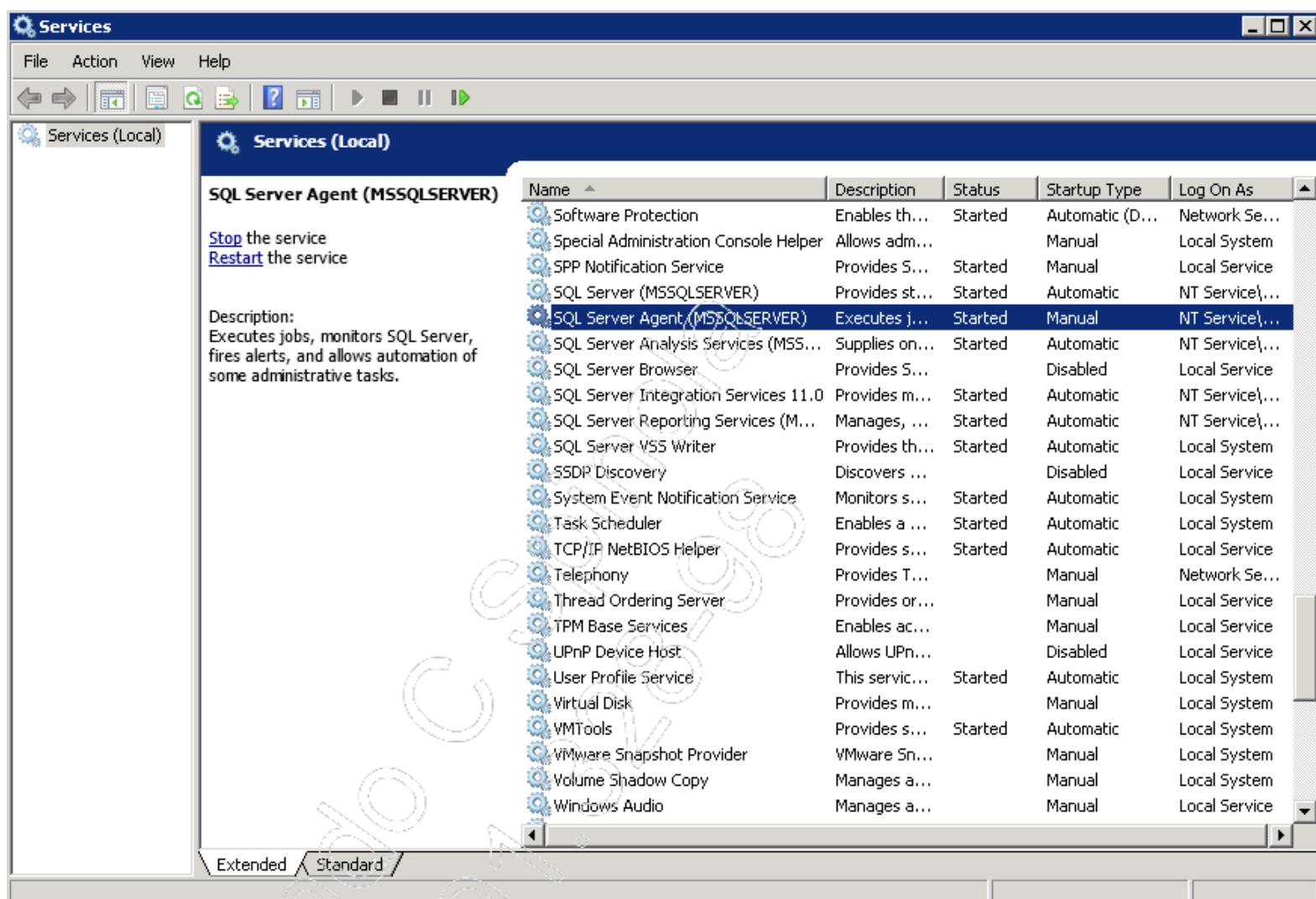
Todos os serviços Windows podem ter e uma grande parte deles já conta com a inicialização automática de serviços. Muitos deles são executados com uma conta chamada de **local system**, indicando que qualquer usuário que conectar no servidor de banco de dados poderá parar e reiniciar o serviço. O usuário precisa ter os seguintes privilégios no sistema operacional:

- **Log on as a service (SeServiceLogonRight)**: Fazer logon como um service (usando credenciais);
- **Replace a process-level token (SeAssignPrimaryTokenPrivilege)**: Substituir um token de nível de processo;
- **Bypass traverse checking (SeChangeNotifyPrivilege)**: Ignorar verificação completa;
- **Adjust memory quotas for a process (SeIncreaseQuotaPrivilege)**: Ajustar cotas de memória para um processo.

Caso esses privilégios não sejam dados ao usuário de inicialização dos serviços, o agendamento de atividades (Jobs) deverá ser feito por um usuário pertencente à FIXED SERVER ROLE **sysadmin**.

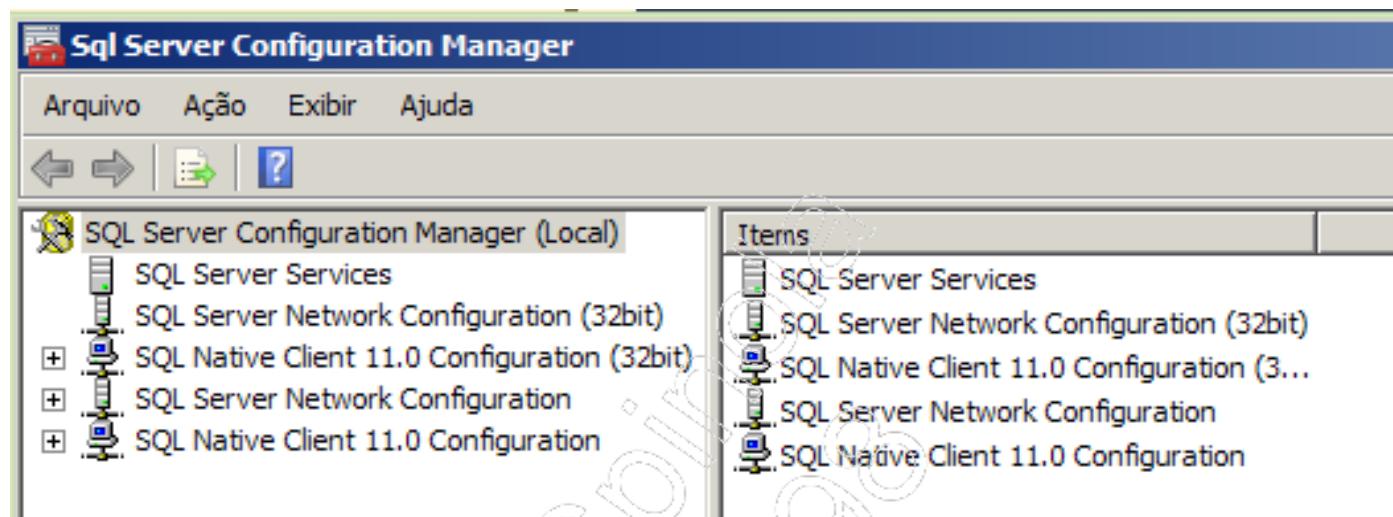
SQL 2014 - Módulo III

Para inicializar ou parar os serviços do SQL Server, inclusive o SQL Server Agent, clique no botão Start, escolha Execute e digite **services.msc**.



Automação de tarefas, alertas e operadores

Os serviços também podem ser manipulados através da ferramenta de gerenciamento de configuração do SQL Server (**SQL Server Configuration Manager**), que pode ser encontrada no menu do SQL Server 2014 ou ainda acessada por meio de **Start / Execute** e, em seguida, digitar **SQLServerManager12.msc**.



Name	State	Start Mode	Log On As	Proce...	Service Type
SQL Server Integration Services 11.0	Stopped	Manual	NT AUTHORITY\NET...	0	
SQL Full-text Filter Daemon Launcher (MSSQLSERVER)	Running	Manual	NT AUTHORITY\LOC...	2956	
SQL Server (MSSQLSERVER)	Running	Automatic	NT AUTHORITY\NET...	1408	SQL Server
SQL Server Analysis Services (MSSQLSERVER)	Stopped	Manual	NT AUTHORITY\NET...	0	Analysis Server
SQL Server Reporting Services (MSSQLSERVER)	Stopped	Manual	NT AUTHORITY\NET...	0	Report Server
SQL Server Browser	Stopped	Other (Boot, Syste...	NT AUTHORITY\LOC...	0	
SQL Server Agent (MSSQLSERVER)	Running	Manual	LocalSystem	3336	SQL Agent

Através do **Configuration Manager**, podemos inicializar e parar as instâncias SQL Server e demais serviços ligados ao banco de dados.

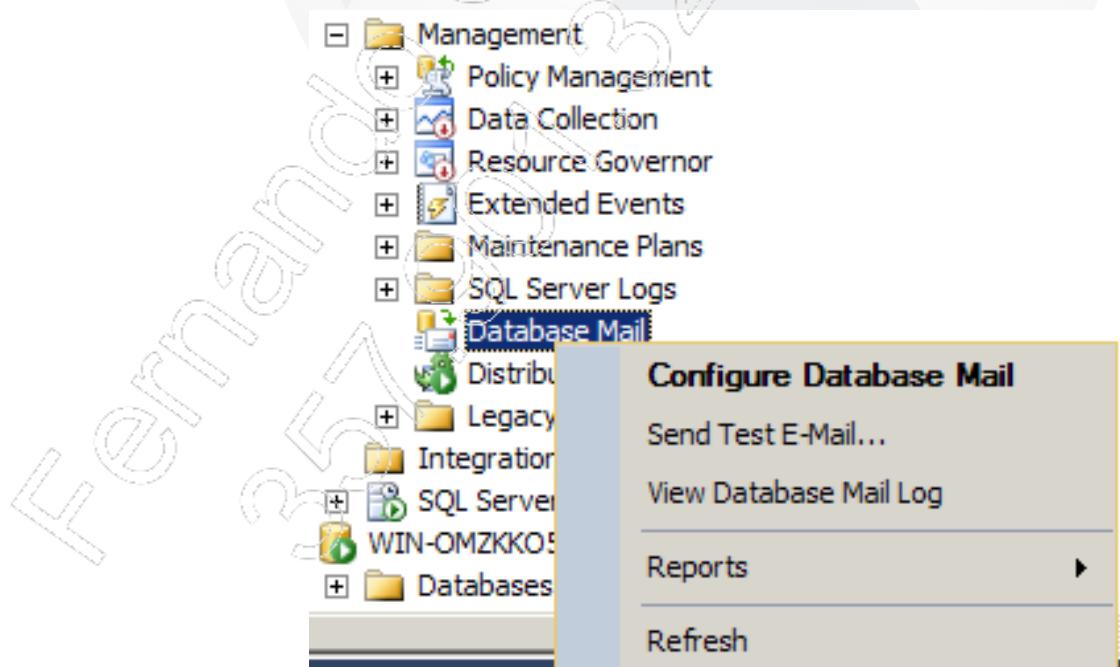
O SQL Server Agent deve ser iniciado após a inicialização da instância SQL e tem vinculação com o serviço de instância. Caso o serviço de instância seja interrompido, o serviço do SQL Server Agent será interrompido antes.

8.3. Configurando o envio de e-mails através do SQL Server

O SQL Server Agent dispõe de uma ferramenta para envio de e-mail chamada **Database Mail**. Ela pode ser configurada para enviar mensagens em situações de agendamento de tarefas, sejam elas bem-sucedidas ou não, e quando há a ocorrência de algum alerta pré-configurado. Além do Database Mail, existe mais uma ferramenta para envio de e-mails chamada **SQL Mail**. Ela oferece o recurso **extended MAPI**, que está em desuso desde a versão 2008, logo não deve mais ser utilizada na versão 2014, exceto em casos de migração de um banco de dados de versão inferior.

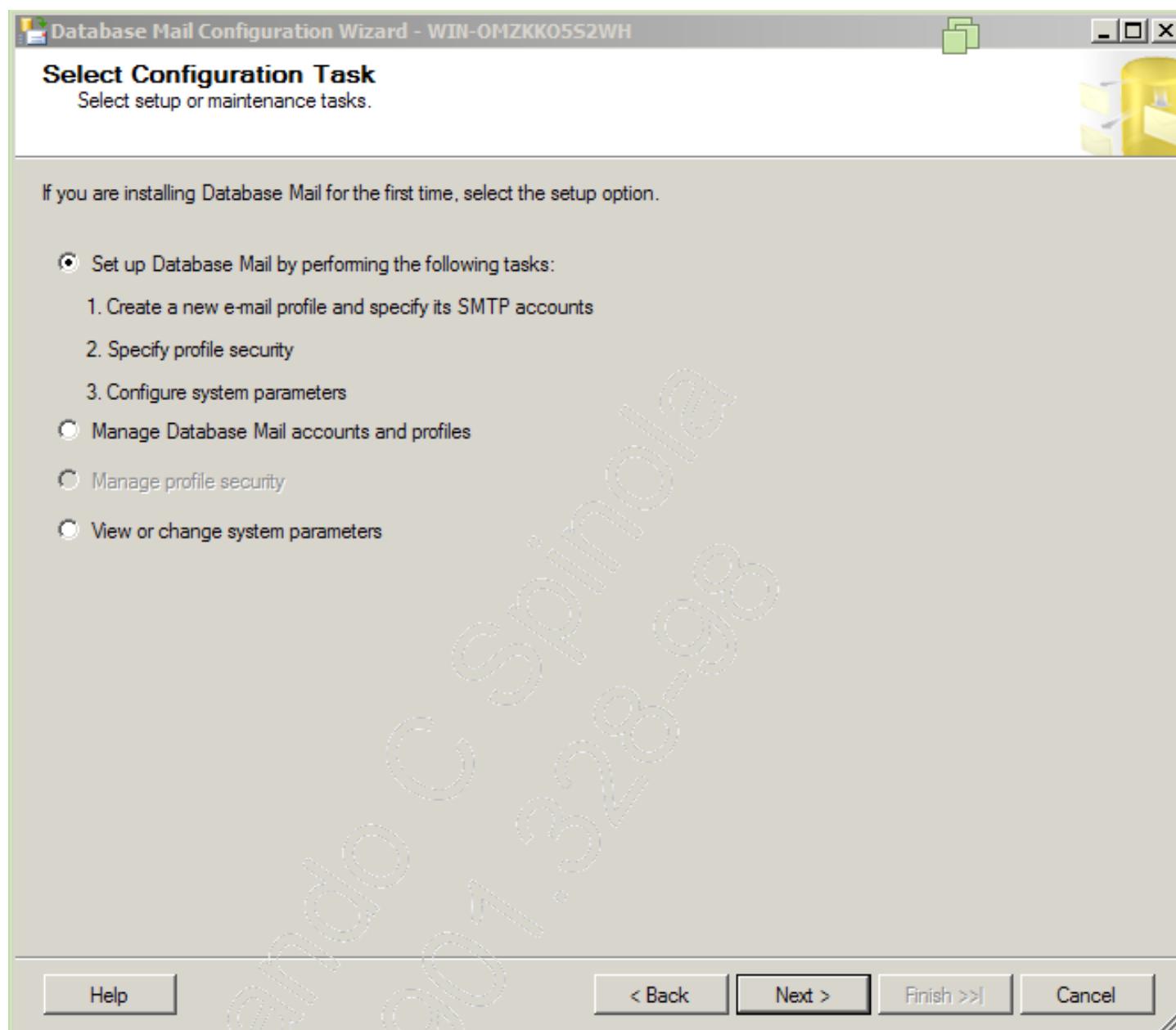
O Database Mail é a solução que permite criar aplicações em banco de dados e, a partir deles, enviar e-mails. Este recurso não está ativado por padrão. Para ativá-lo, devemos usar o assistente de configuração do Database Mail ou a ferramenta **SQL Server Surface Area**.

A imagem a seguir mostra o assistente para configuração do Database Mail. Selecione a opção **Configure Database Mail**:

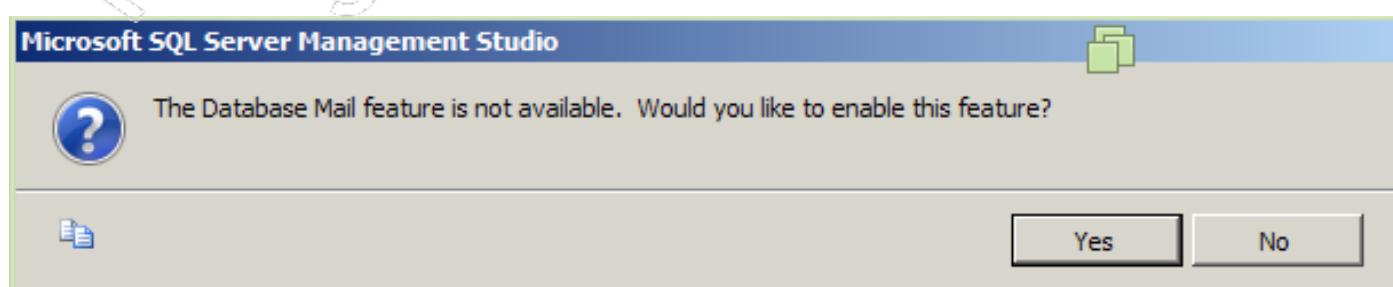


Automação de tarefas, alertas e operadores

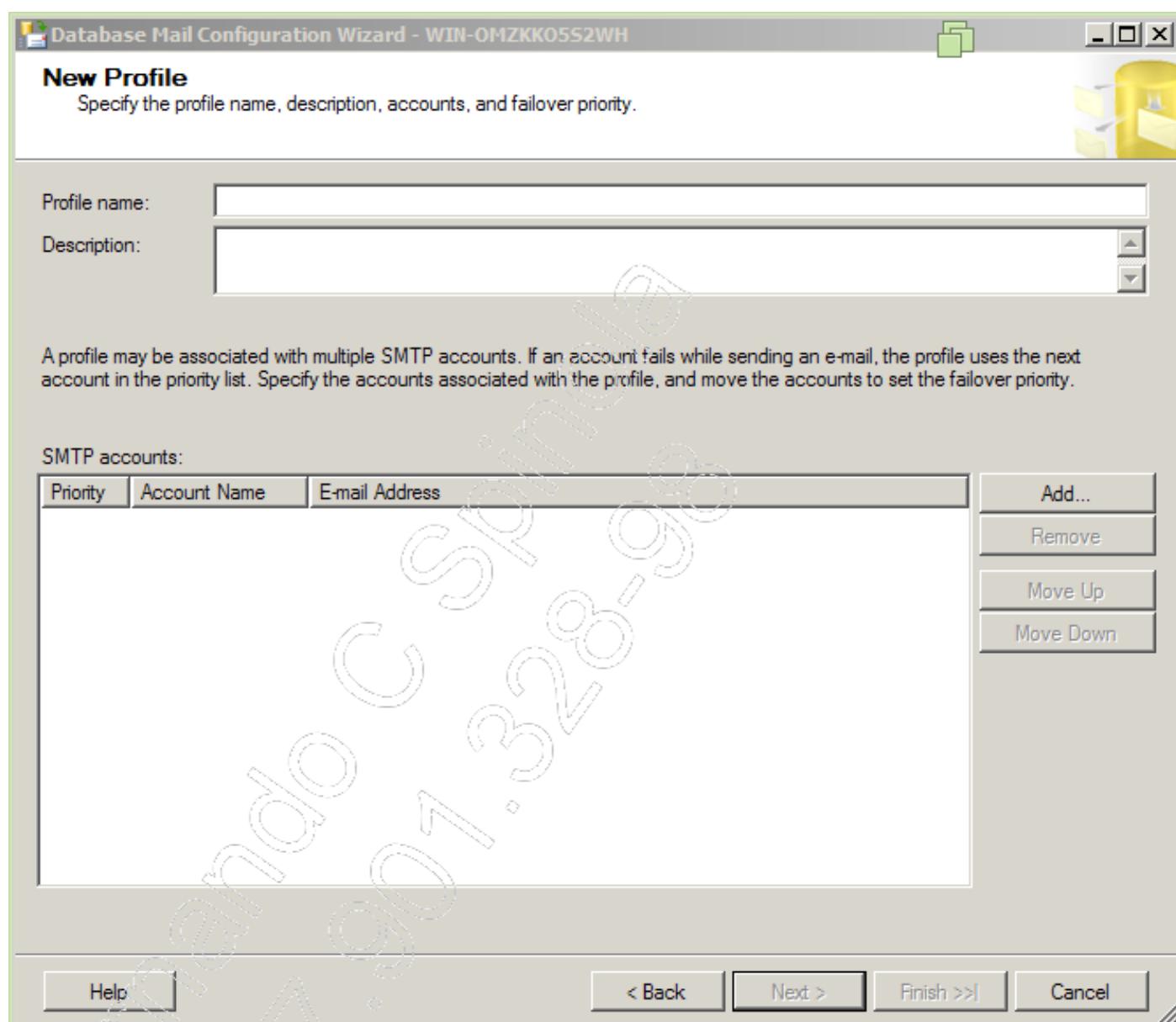
Esta janela exibe a configuração do Database Mail inicialmente (**Set up Database Mail by performing the following tasks**):



O assistente identifica que o recurso do Database Mail está desligado. Selecione o botão **Yes** para habilitar o recurso:

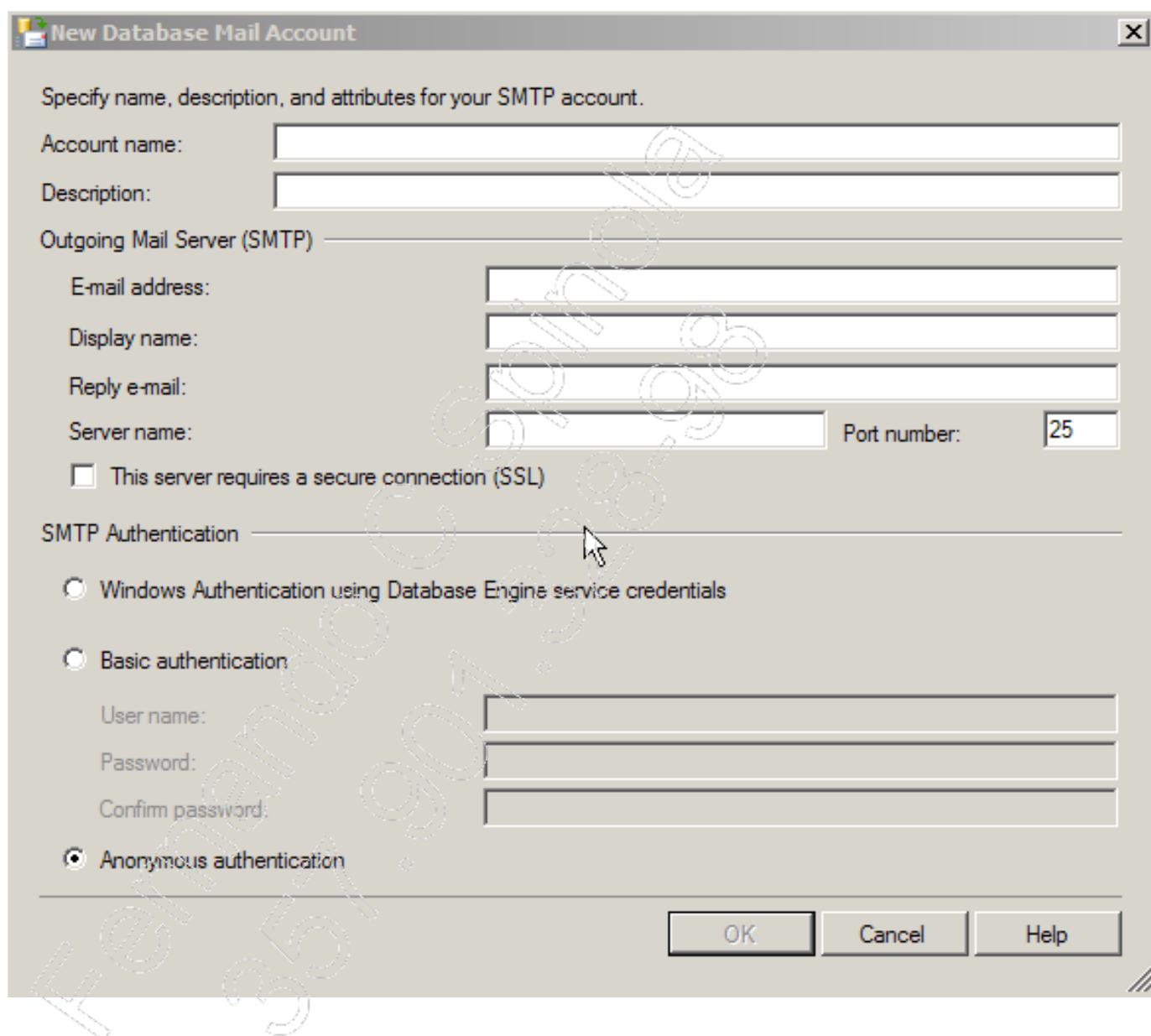


Defina o nome do perfil (**Profile name**) e adicione uma conta SMTP através do botão **Add...**:



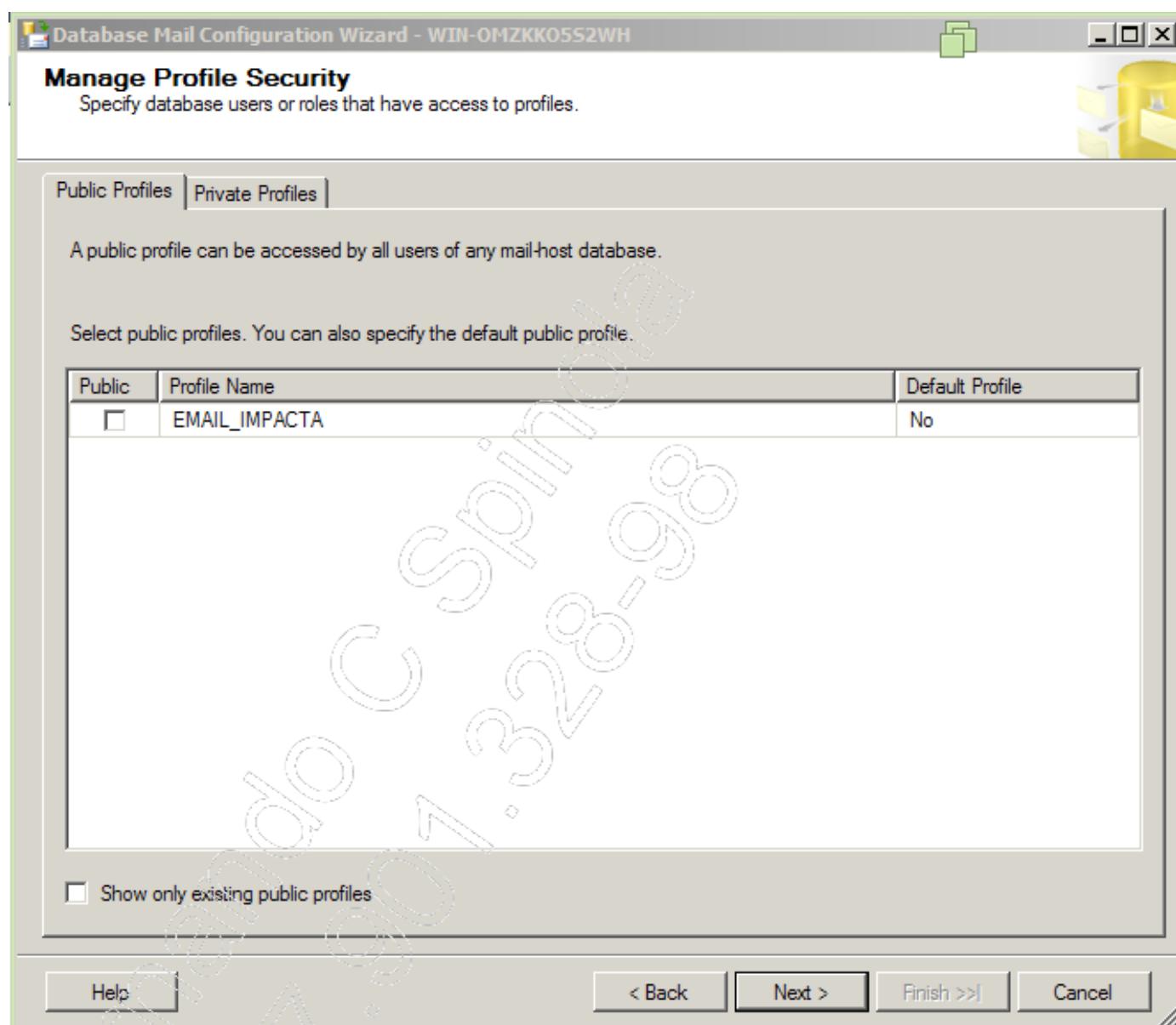
Especifique o nome da conta SMTP (**Account name**), o e-mail de envio (**E-mail Address**), o nome que deverá aparecer no e-mail (**Display name**), o e-mail para retorno (**Reply e-mail**), o nome ou IP do servidor (**Server name**) e o número da porta (**Port number**).

Para autenticação SMTP, pode-se usar autenticações anônimas (**Anonymous authentication**), quando o recurso for suportado. A autenticação também pode ser feita usando o **Windows Authentication using Database Engine service credentials**. A opção de autenticação básica é aquela em que se deve informar o usuário do Windows e sua senha.

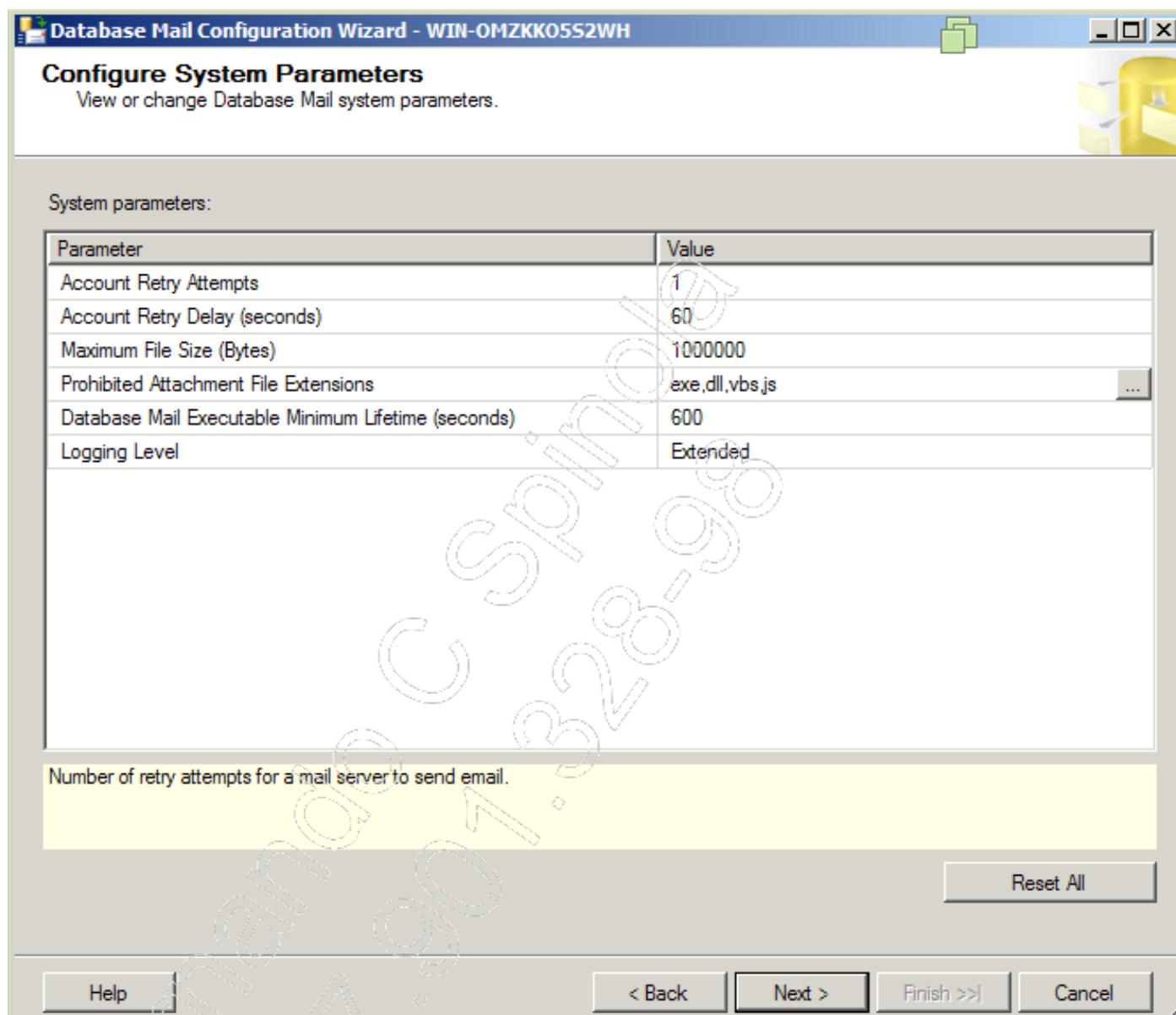


SQL 2014 - Módulo III

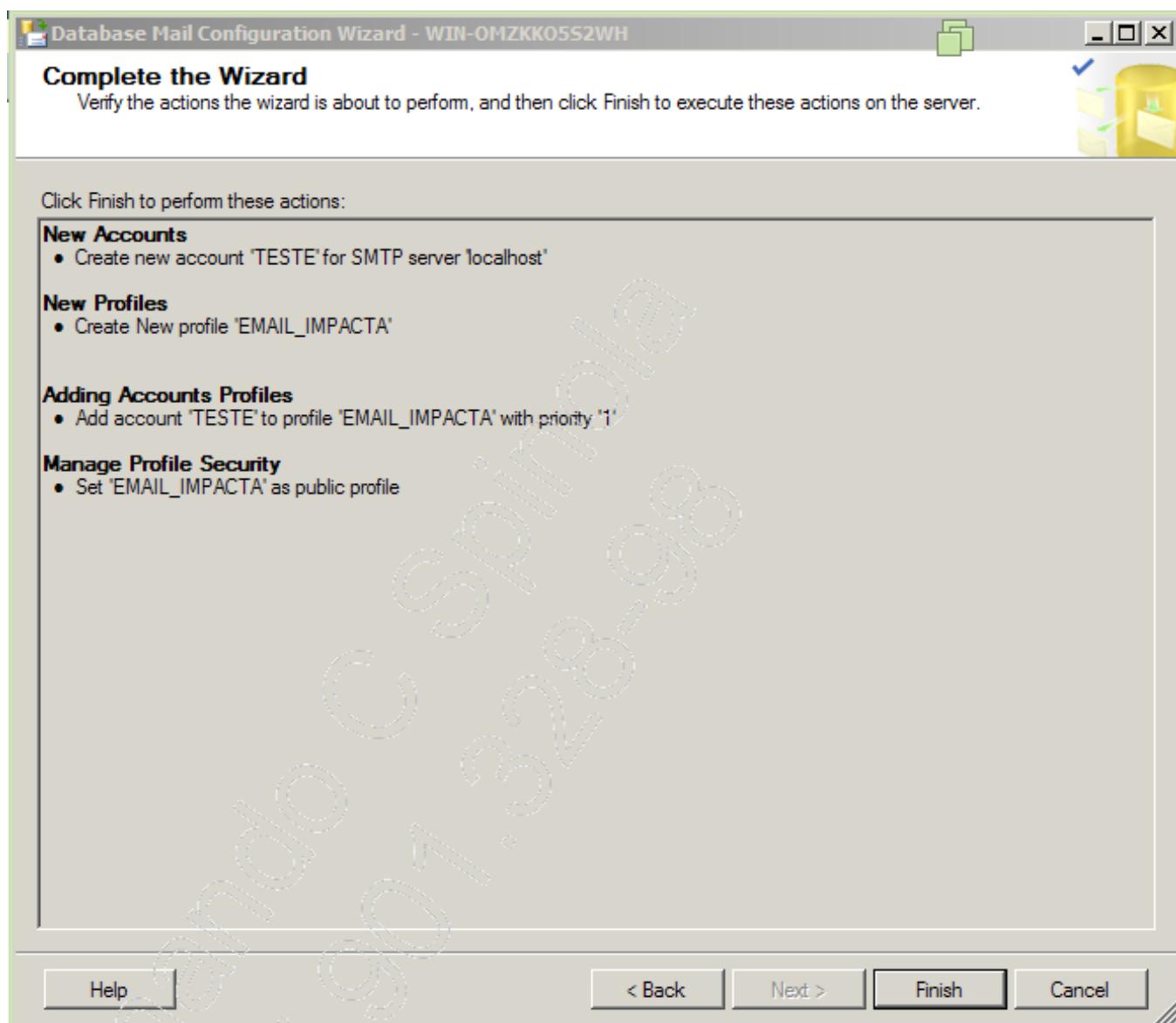
Selecione um perfil, marcando a opção **Public** e clicando no botão **Next**.



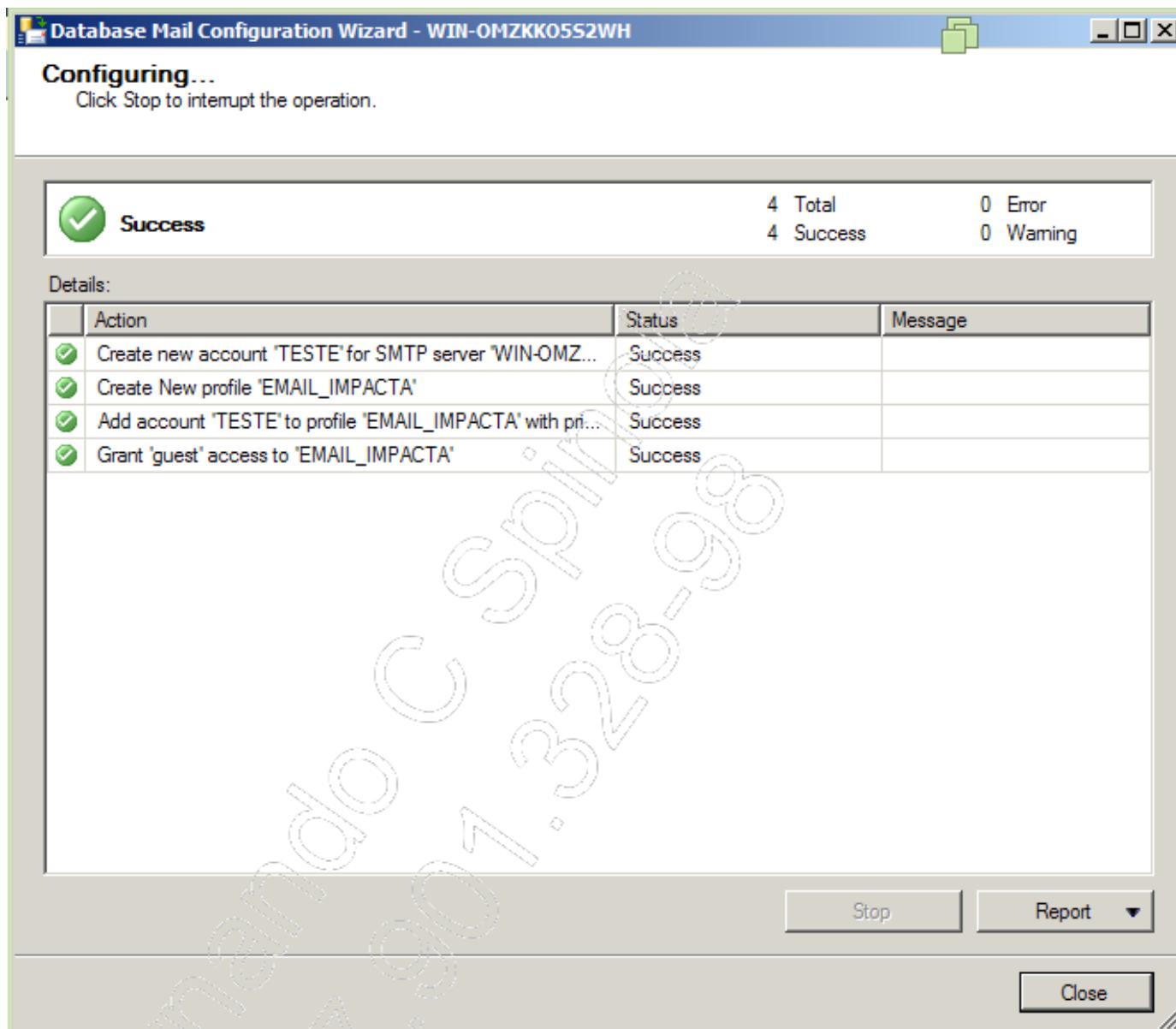
Clique no botão **Next** na janela que será exibida:



Clique no botão **Finish** na próxima janela:



A última tela do assistente mostra o status de configuração do Database Mail:



Os procedimentos **SP_SEND_DBMAIL** e **XP_SENDMAIL** (até a versão 2008 R2) podem ser usados para o envio de e-mails quando utilizamos o Database Mail.

SQL 2014 - Módulo III

A sintaxe do procedimento **SP_SEND_DBMAIL** é:

```
sp_send_dbmail [ [ @profile_name = ] 'profile_name' ]
    [ , [ @recipients = ] 'recipients [ ; ...n ]' ]
    [ , [ @copy_recipients = ] 'copy_recipient [ ; ...n ]' ]
    [ , [ @blind_copy_recipients = ] 'blind_copy_recipient [ ;
...n ]' ]
    [ , [ @from_address = ] 'from_address' ]
    [ , [ @reply_to = ] 'reply_to' ]
    [ , [ @subject = ] 'subject' ]
    [ , [ @body = ] 'body' ]
    [ , [ @body_format = ] 'body_format' ]
    [ , [ @importance = ] 'importance' ]
    [ , [ @sensitivity = ] 'sensitivity' ]
    [ , [ @file_attachments = ] 'attachment [ ; ...n ]' ]
    [ , [ @query = ] 'query' ]
    [ , [ @execute_query_database = ] 'execute_query_database' ]
    [ , [ @attach_query_result_as_file = ] attach_query_result_
as_file ]
    [ , [ @query_attachment_filename = ] query_attachment_
filename ]
    [ , [ @query_result_header = ] query_result_header ]
    [ , [ @query_result_width = ] query_result_width ]
    [ , [ @query_result_separator = ] 'query_result_separator' ]
    [ , [ @exclude_query_output = ] exclude_query_output ]
    [ , [ @append_query_error = ] append_query_error ]
    [ , [ @query_no_truncate = ] query_no_truncate ]
.....[ , [ @query_result_no_padding = ] @query_result_no_padding ]
    [ , [ @mailitem_id = ] mailitem_id ] [ OUTPUT ]
```

Em que:

- **[@profile_name=] 'profile_name'**: Nome do perfil que enviará a mensagem. O **profile_name** possui o padrão NULL. Caso não seja especificado, será utilizado o perfil do usuário que estiver acionando a stored procedure **SP_SEND_DBMAIL**. Se nenhum perfil estiver associado ao usuário, será utilizado o perfil público para o banco de dados **msdb**. Se nenhuma das duas opções anteriores existir, o **profile_name** deverá ser obrigatoriamente informado;

- [@recipients=] ‘recipients’: Define uma lista delimitada por ponto e vírgula de endereços de e-mail para os quais a mensagem será enviada. Esta lista é opcional, pois, caso sejam especificados apenas @copy_recipients ou @blind_copy_recipients, o e-mail também poderá ser enviado. Se nenhum dos parâmetros tiver sido especificado, a stored procedure **sp_send_dbmail** retornará um erro;
- [@copy_recipients=] ‘copy_recipients’: Lista de endereços de e-mail, separados por ponto e vírgula, que entrará na cópia da mensagem. O tipo da lista de destinatários de cópia é **varchar(max)**. Trata-se de um parâmetro opcional, porém, para que **sp_send_dbmail** não retorne um erro, devemos determinar pelo menos @recipients, @copy_recipients ou @blind_copy_recipients;
- [@blind_copy_recipients=] ‘blind_copy_recipients’: Lista de e-mails, demarcada por ponto e vírgula, que entrará na cópia oculta da mensagem. Trata-se de um parâmetro opcional, porém, para que **sp_send_dbmail** não retorne um erro, devemos determinar pelo menos @recipients, @copy_recipients ou @blind_copy_recipients;
- [@from_address=] ‘from_address’: Parâmetro que indica o endereço de e-mail de origem da mensagem. Caso não seja especificado, será atribuído ao perfil do usuário. Se este perfil não existir, o e-mail será atribuído ao perfil público do banco de dados **msdb**;
- [@reply_to=] ‘reply_to’: É o parâmetro para envio de resposta ao e-mail enviado. O padrão será NULL se não definirmos nenhum parâmetro;
- [@subject=] ‘subject’: Assunto da mensagem (opcional). Pode ocupar até 255 caracteres **NVARCHAR**. Seu valor padrão é **SQL Server Message**;
- [@body=] ‘body’: Trata-se do corpo da mensagem do e-mail. Utiliza o tipo **NVARCHAR(max)** e seu padrão é NULL;
- [@body_format=] ‘body_format’: Trata-se do formato do corpo da mensagem e tem como padrão NULL. Este parâmetro pode conter o valor **TEXT** (Padrão) ou o valor **HTML**;

- [**@importance=**] ‘**importance**’: É o grau de relevância da mensagem. Aceita os valores Low, Normal (Padrão) e High;
- [**@sensitivity=**] ‘**sensitivity**’: Indica a qualidade da mensagem. Aceita os valores Normal (Padrão), Personal, Private e Confidential;
- [**@file_attachments=**] ‘**file_attachments**’: Lista contendo os nomes de arquivos. Caso haja mais de um arquivo, os nomes devem ser separados por ponto e vírgula. O limite para cada arquivo é de 1 MB;
- [**@query=**] ‘**query**’: Consulta que pode estar vinculada ao e-mail, ou seja, permite que o resultado dessa consulta seja incorporado ao e-mail. Este parâmetro é opcional e seu padrão é NULL;
- [**@execute_query_database=**] ‘**execute_query_database**’: Indica o banco de dados a ser utilizado, somente válido caso seja especificado o parâmetro **@query**;
- [**@attach_query_result_as_file=**] **attach_query_result_as_file**: Indica se deseja anexar o resultado da consulta como um arquivo ao invés de anexá-lo ao corpo do e-mail. Esse parâmetro só é aplicável se **@query** for especificado. Indica os valores 0 e 1. Caso seja especificado 1, o parâmetro **@query_attachment_filename** deverá ser informado. O parâmetro é NULL;
- [**@query_attachment_filename=**] **query_attachment_filename**: Indica nome do arquivo da consulta. Deve ser especificado quando o valor de **@attach_query_result_as_file** for igual a 1. O padrão para este parâmetro é NULL;
- [**@query_result_header=**] **query_result_header**: Indica se os cabeçalhos de coluna estão incluídos nos resultados da consulta. Se for especificado o valor 1, o cabeçalho da consulta será apresentado. Ao definir o valor 0 (zero), as colunas não serão exibidas na consulta;
- [**@query_result_width =**] **query_result_width**: Trata-se da largura de linha para formatação do arquivo ou do corpo do e-mail. Este é o número definido para o tamanho da linha dos resultados da consulta. O parâmetro **query_result_width** tem como padrão 256 caracteres, porém, pode-se utilizar valores entre 10 e 32767. A aplicação desse parâmetro depende da definição de **@query**;

- [**@query_result_separator=**] ‘**query_result_separator**’: Especifica o caractere que separará as colunas na consulta. O separador é definido por um único caractere e utiliza ‘ ‘ (espaço) como padrão;
- [**@exclude_query_output=**] **exclude_query_output**: Esta opção inibe a impressão da consulta no e-mail. Ao definirmos 0 (zero) e caso a consulta tenha sido especificada no parâmetro **@query**, seu resultado será impresso. Se esse parâmetro for definido como 1, nenhuma mensagem de execução de consulta no console será impressa pela execução do procedimento armazenado **sp_send_dbmail**.

Vejamos um exemplo de uso de **SP_SEND_DBMAIL**:

```
EXEC msdb.dbo.sp_send_dbmail
    @profile_name = 'Administrador DB_MAIL',
    @recipients = 'dba@xpto.com.br',
    @body = 'Procedimento encerrado com sucesso',
    @subject = 'Procedimento de Compressão dos DataFiles' ;
```

O procedimento **XP_SENDMAIL** possui a seguinte sintaxe:

```
xp_sendmail { [ @recipients= ] 'recipients [ ;...n ]' }
    [ , [ @message= ] 'message' ]
    [ , [ @query= ] 'query' ]
    [ , [ @attachments= ] 'attachments [ ;...n ]' ]
    [ , [ @copy_recipients= ] 'copy_recipients [ ;...n ]'
    [ , [ @blind_copy_recipients= ] 'blind_copy_recipients [
;...n ]'
    [ , [ @subject= ] 'subject' ]
    [ , [ @type= ] 'type' ]
    [ , [ @attach_results= ] 'attach_value' ]
    [ , [ @no_output= ] 'output_value' ]
    [ , [ @no_header= ] 'header_value' ]
    [ , [ @width= ] width ]
    [ , [ @separator= ] 'separator' ]
    [ , [ @echo_error= ] 'echo_value' ]
    [ , [ @set_user= ] 'user' ]
    [ , [ @dbuse= ] 'database' ]
```

Em que:

- [**@recipients=**] ‘recipients [;... n]’: É a lista de destinatários de e-mail. Caso tenhamos mais do que um destinatário, eles deverão ser separados pelo caractere ponto e vírgula;
- [**@message=**] ‘message’: Trata-se da mensagem que será enviada por e-mail. Ela pode conter até 8.000 bytes;
- [**@query=**] ‘query’: Consulta que poderá ser realizada no SQL Server e ser vinculada ao e-mail informado. Este parâmetro é opcional. A stored procedure **xp_sendmail** usa uma conexão de saída que não sofre com os bloqueios mantidos pelo cliente que emite a solicitação **xp_sendmail**. Isso torna mais fácil o uso de **xp_sendmail** em gatilhos. A instrução **query**, entretanto, não pode se referir às tabelas temporárias inseridas e excluídas, porque elas só estão disponíveis em um gatilho. O tamanho máximo é de 8.000 bytes;
- [**@attachments=**] ‘attachments [;... n]’: Lista de arquivos a serem anexados ao e-mail. Caso tenhamos mais de um arquivo, eles deverão ser separados por ponto e vírgula. O padrão é NULL;
- [**@copy_recipients=**] ‘copy_recipients [;... n]’: Lista que identifica os e-mails que deverão ser enviados em cópia. Este parâmetro é opcional e tem como padrão NULL. Caso exista mais de um e-mail a ser enviado em cópia, os endereços deverão ser separados por ponto e vírgula;
- [**@blind_copy_recipients=**] ‘blind_copy_recipients[;... n]’: Lista que identifica os e-mails que deverão ser enviados em cópia oculta. Este parâmetro é opcional e tem como padrão NULL. Caso exista mais de um e-mail a ser enviado em cópia, os endereços deverão ser separados por ponto e vírgula;
- [**@subject=**] ‘subject’: Assunto do e-mail. Se este parâmetro não for especificado, o padrão será definido como **SQL Server Message**;
- [**@type=**] ‘type’: Trata-se do tipo da mensagem de e-mail baseada na definição de e-mail MAPI:

IP[M|C].Vendorname.subclass

Se o valor para **type** for NULL, **xp_sendmail** usará um tipo de mensagem de IPM. Os tipos de mensagem que iniciam com IPM aparecem na caixa de entrada do cliente de e-mail e são encontrados ou lidos pela stored procedure **xp_findnextmsg**. Os tipos de mensagem que iniciam com IPC não são visíveis na caixa de entrada do cliente de e-mail e devem ser encontrados ou lidos pela configuração definida neste parâmetro. O padrão para este parâmetro é NULL;

- [**@attach_results=**] ‘**attach_value**’: Parâmetro que indica que o resultado de uma consulta deve ser anexado como arquivo ao e-mail. Se o parâmetro **@attachments** não for NULL e o parâmetro **@attach_results** for TRUE, o primeiro nome de arquivo no parâmetro **@attachments** é usado como o nome de arquivo para os resultados. O padrão é falso, indicando que o conjunto de resultados é anexado à mensagem;
- [**@no_output=**] ‘**output_value**’: Indica se a sessão do usuário que enviou o e-mail deve receber uma saída ou não. O padrão é FALSE;
- [**@no_header=**] ‘**header_value**’: Indica se o cabeçalho deverá ser enviado ou não. O padrão é FALSE, indicando que esse parâmetro deverá ser enviado;
- [**@width=**] **width**: Parâmetro que indica o tamanho da largura do registro a ser impresso. A largura padrão é de 80 caracteres;
- [**@separator=**] ‘**separator**’: Indica o caractere a ser utilizado para separação das colunas em uma consulta. O padrão é o espaço em branco, mas podemos utilizar vírgula ou ainda o ponto e vírgula para geração de arquivos;
- [**@echo_error=**] ‘**echo_value**’: Captura qualquer mensagem de erro lançada na execução da consulta. Isso permite que erros que possam ocorrer durante a execução da consulta sejam lançados no e-mail ao invés do arquivo **ERRORLOG**. Indica se o e-mail foi enviado com sucesso ou não. O retorno independe de existirem erros na consulta ou não;
- [**@set_user=**] ‘**user**’: Indica com qual usuário a consulta deverá ser realizada. Este parâmetro é opcional e seu padrão é NULL;
- [**@dbuse=**] ‘**database**’: Nome do banco de dados a ser utilizado. É recomendável que este parâmetro seja indicado.

8.4. Configurando tarefas (Jobs)

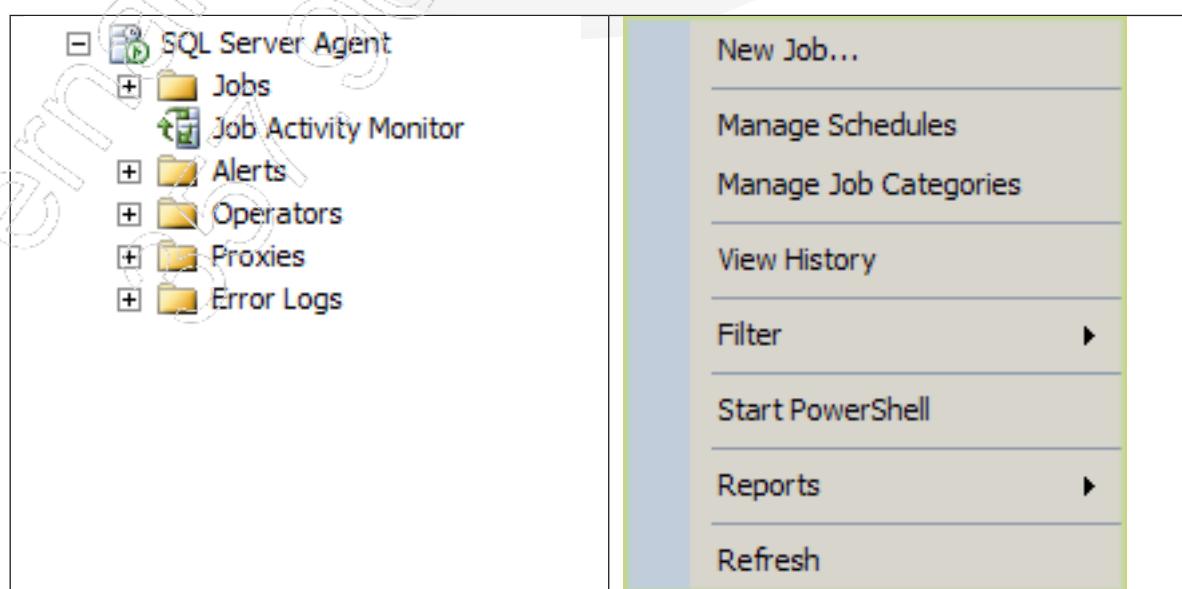
Tarefas (Jobs) são ações agendadas que envolvem o uso de comandos visando uma finalidade. Os comandos suportados em Jobs são SQL, T-SQL, Powershell, ActiveX e comandos batch no sistema operacional.

É importante que as tarefas sejam claras e que tenham um fluxo de execução, seja no que diz respeito aos scripts ou mesmo à execução de procedimentos (Procedures) do SQL Server.

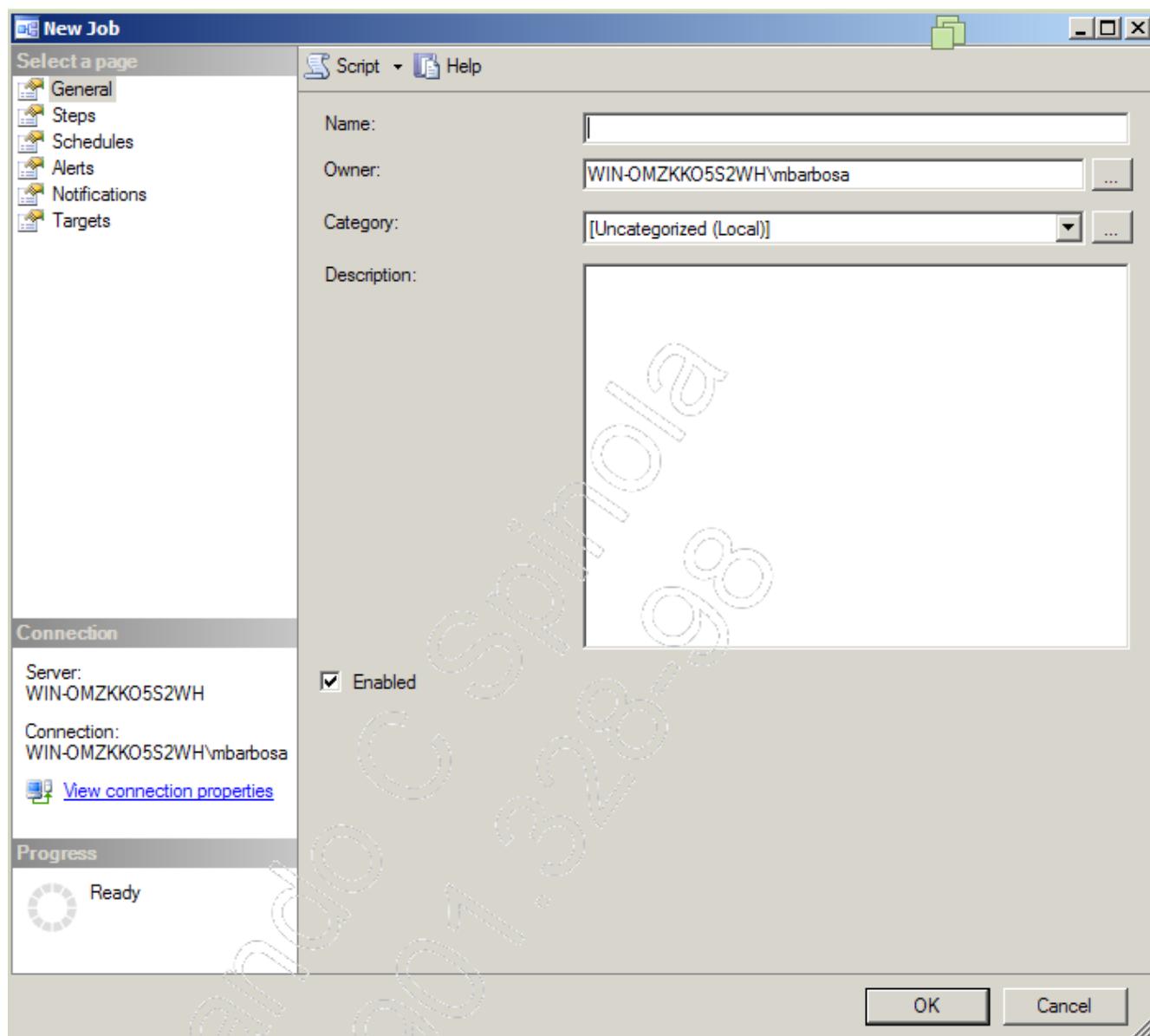
Além disso, é importante observar os seguintes fatores:

- Definir o proprietário da tarefa. Por padrão, o dono tarefa é quem a cria;
- Definir o local de execução da tarefa se será no local ou em outro servidor;
- Verificar se a tarefa está habilitada. Por padrão, ao criarmos uma tarefa, ela ficará habilitada;
- Procurar criar categorias de tarefas, a fim de melhorar o gerenciamento delas. Isso pode ser importante à medida que o número de tarefas aumenta em um mesmo servidor.

Vejamos a opção para criar tarefas no SSMS. Primeiramente, selecione a opção **New Job....**

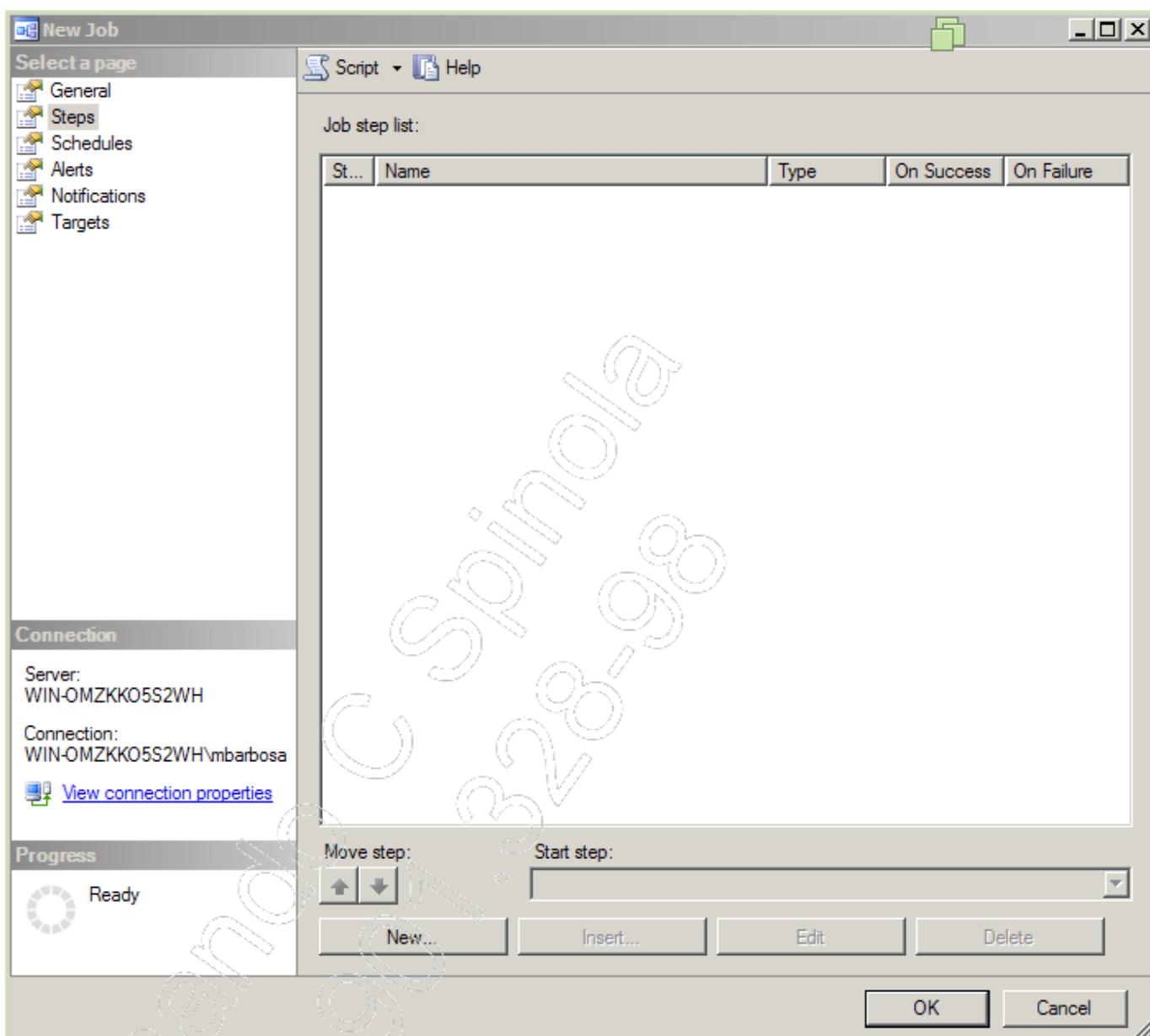


Defina o nome (**Name**), o proprietário (**Owner**) e a categoria do Job (**Category**). Em seguida, escreva o propósito desse Job em **Description**. Caso queira deixá-lo habilitado, marque a opção **Enabled**.

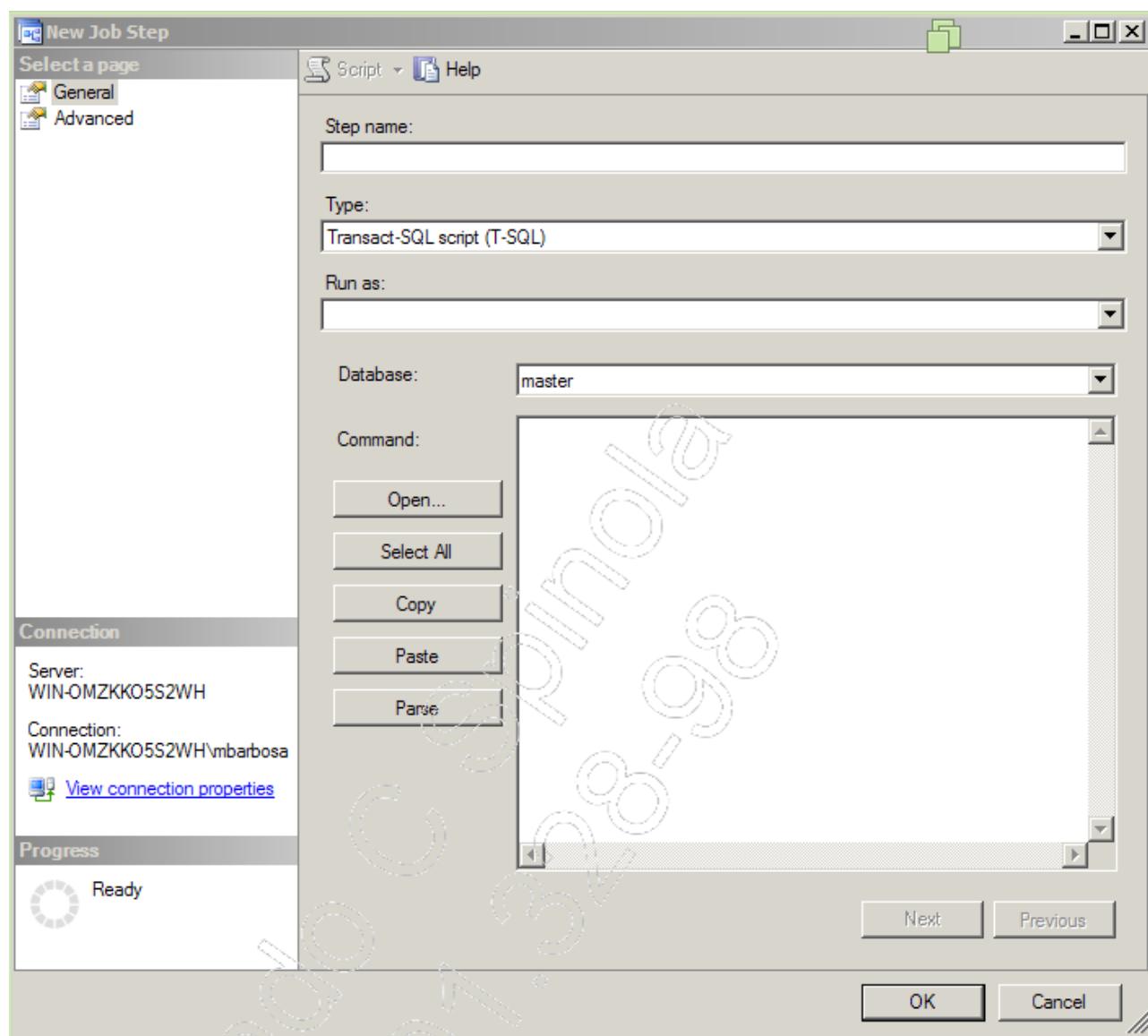


SQL 2014 - Módulo III

Selecione, na lateral esquerda da tela, a opção **Steps**. Em seguida, clique no botão **New...** para inserir um passo na tarefa, lembrando que uma tarefa poderá ter quantos passos forem necessários.



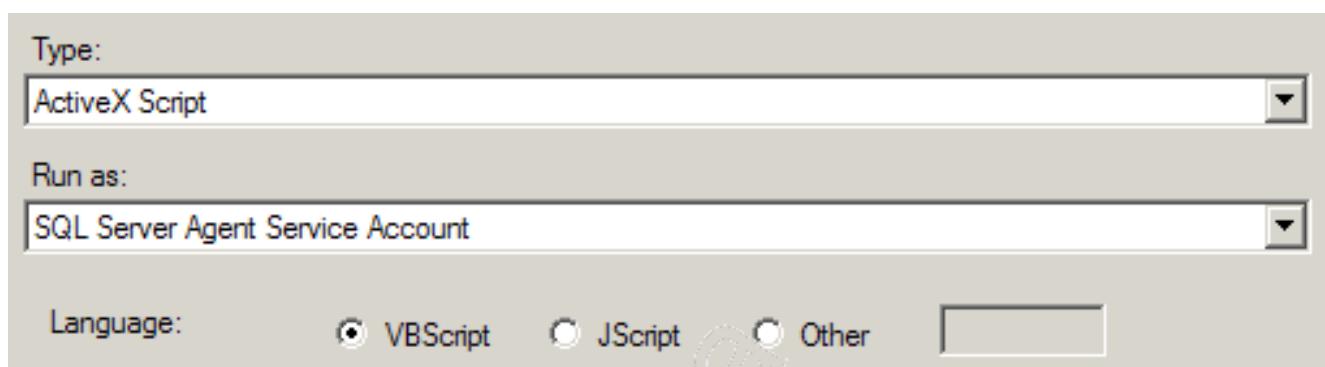
Defina o nome do passo (**Step name**) e o tipo de tarefa a ser realizado (**Type**), conforme a lista a seguir:



ActiveX Script
Operating system (CmdExec)
PowerShell
Replication Distributor
Replication Merge
Replication Queue Reader
Replication Snapshot
Replication Transaction-Log Reader
SQL Server Analysis Services Command
SQL Server Analysis Services Query
SQL Server Integration Services Package
Transact-SQL script (T-SQL)

SQL 2014 - Módulo III

Caso defina a opção **ActiveX Script**, a tela mostrará as seguintes opções:



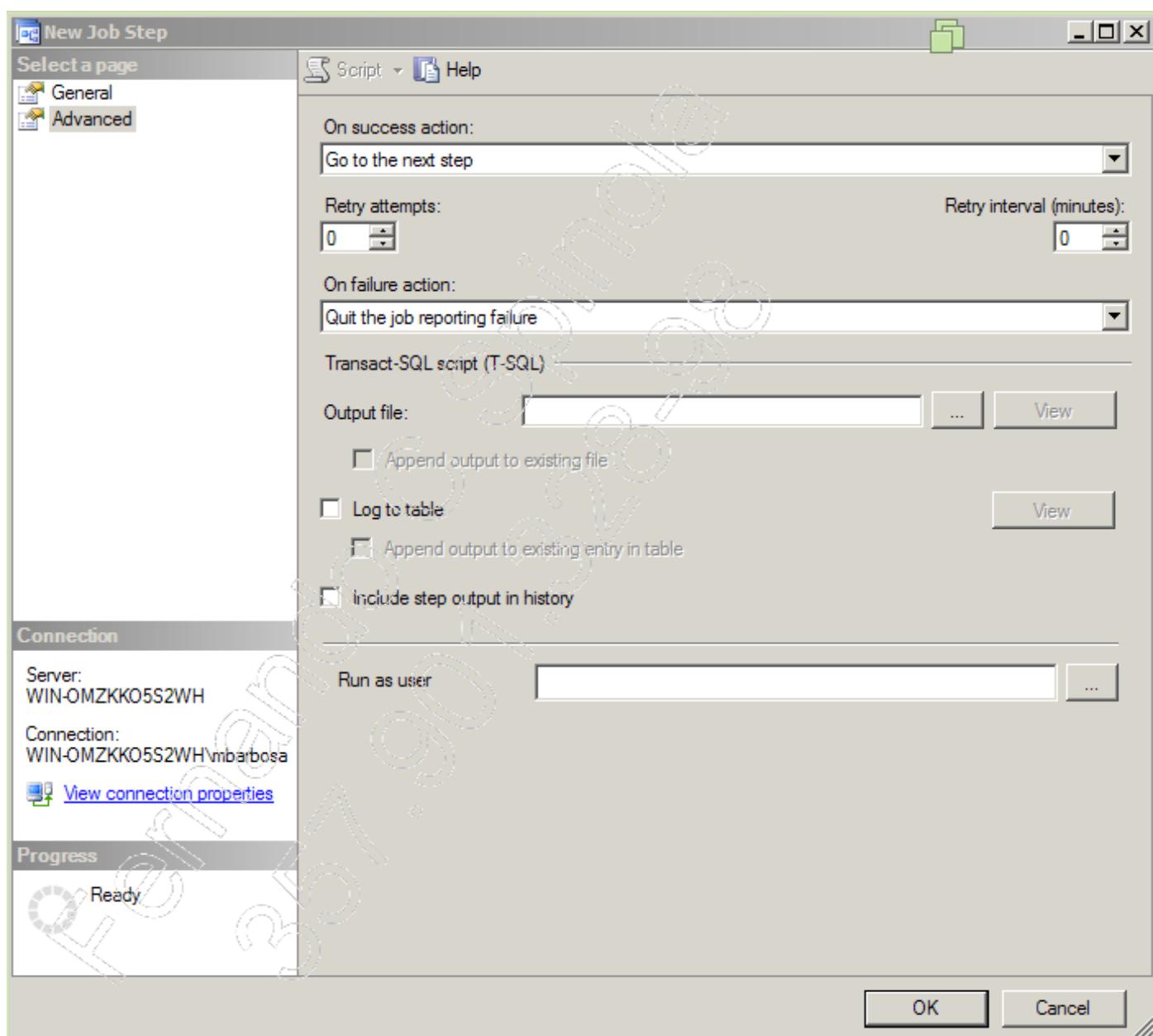
Defina a linguagem a ser utilizada pelo ActiveX. Caso a opção escolhida seja **CmdExec**, a tela mostrará as seguintes opções:



Informe como a tarefa será executada no sistema operacional e defina o código de saída do script batch que será executado em caso de sucesso desse script. O padrão nesses casos é zero.

Selecione a opção **Database** (quando disponível) para indicar em qual banco de dados a tarefa será executada, ou ainda a opção **Server** (quando disponível). Neste caso, indique o nome do servidor em que a operação será executada.

Conforme o tipo escolhido, digite o comando (**Command**) compatível. Selecione, na lateral esquerda da tela, a opção avançada (**Advanced**) para determinar o comportamento do passo em caso de sucesso (**On success action**) ou em caso de falha (**On failure action**). Em caso de falha, pode-se ainda determinar a quantidade de novas tentativas (**Retry attempts**) e quantos minutos elas vão demorar para serem refeitas.

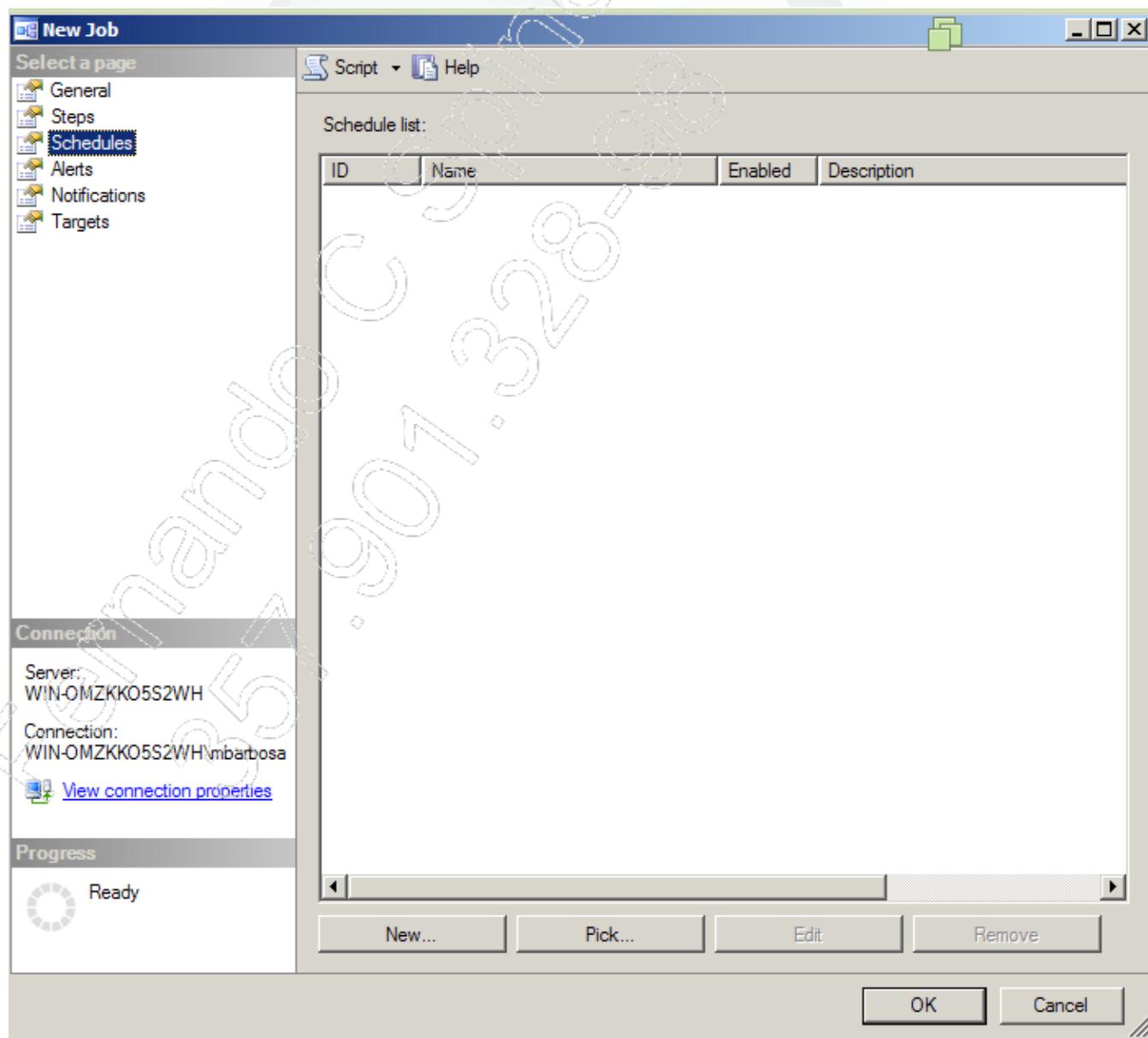


Para os casos de falhas, pode-se ainda vincular um arquivo de saída da execução do armazenamento de informações (**Output file**). Ainda é possível registrar o log de execução do passo em uma tabela (**Log to table**).

Caso queira armazenar o resultado de saída do passo, é possível usar a opção **Include step output in history**. Para finalizar, pode-se executar o passo como outro usuário diferente do dono da tarefa.

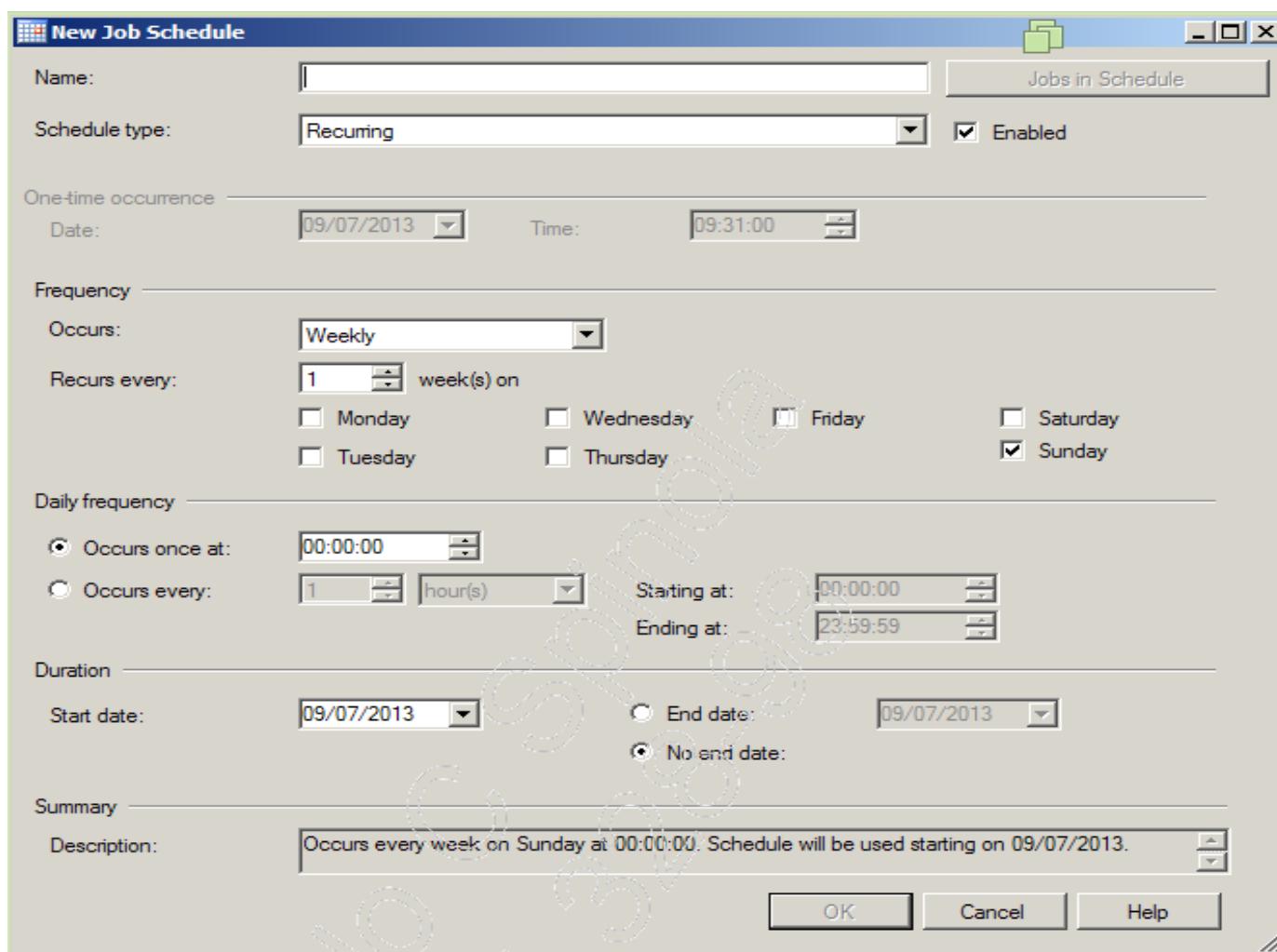
- **Executando uma tarefa agendada**

As tarefas geralmente podem ser agendadas pela opção **Schedules**. Selecione o botão **New...**, para agendá-las em um horário não utilizado anteriormente, ou a opção **Pick...**, para selecionar um horário previamente utilizado por outras tarefas.



Automação de tarefas, alertas e operadores

Informe o nome do agendamento (**Name**), que não deve ter nenhuma relação com o nome do passo ou da tarefa. Recomenda-se dar nomes relativos ao tempo e à recorrência da ação, por exemplo, **Diário por hora**.



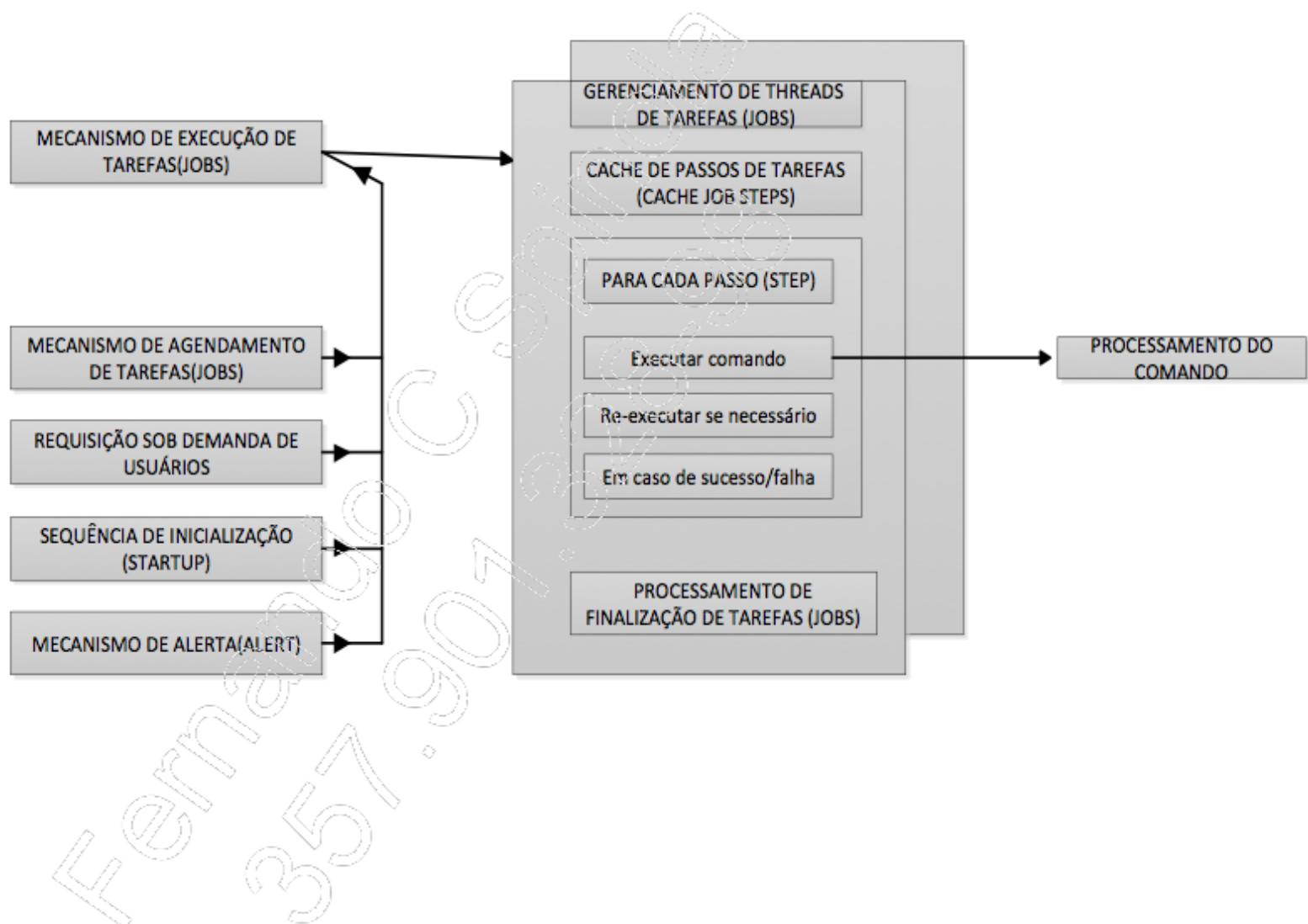
Informe o tipo de agendamento (**Schedule type**), que pode ser: recorrente (**Recurring**), executado somente uma vez (**One time**), iniciado cada vez que o SQL Server Agent for inicializado (**Starts automatically when SQL Server agent starts**), ou quando o processador ficar inativo (**Idle**).

Caso a opção seja recorrente (**Recurring**), deve-se determinar a frequência de ocorrência (**Occurs**). Pode ser diária (**Daily**), semanal (**Weekly**), mensal (**Monthly**) ou anual (**Annually**). Para cada uma dessas opções, deve-se determinar o intervalo (**Recurs every**), que pode indicar o dia desejado, por exemplo, segunda-feira (**Monday**), terça-feira (**Tuesday**), quarta-feira (**Wednesday**), quinta-feira (**Thursday**), sexta-feira (**Friday**), sábado (**Saturday**) e domingo (**Sunday**).

Deve-se ainda indicar o horário para a execução da tarefa, caso seja executada uma única vez (**Occurs once at**). Se for executada várias vezes no período (**Occurs every**), deve-se indicar a faixa de horário em que isso ocorrerá.

Para encerrar, podemos definir a duração da atividade indicando a data de execução da primeira vez (**Start date**) e a data final (**End date**), porém, se a atividade for contínua, pode-se selecionar a opção sem data final (**No end date**).

As tarefas devem seguir o seguinte fluxo de execução:



8.5. Configurando operadores (Operators)

Operadores para o SQL Server são usuários que têm a finalidade de receber notificações e e-mails. O método de notificação é um critério que o administrador pode definir.

Operadores são importantes para criação de alertas, que podem estar vinculados a tarefas.

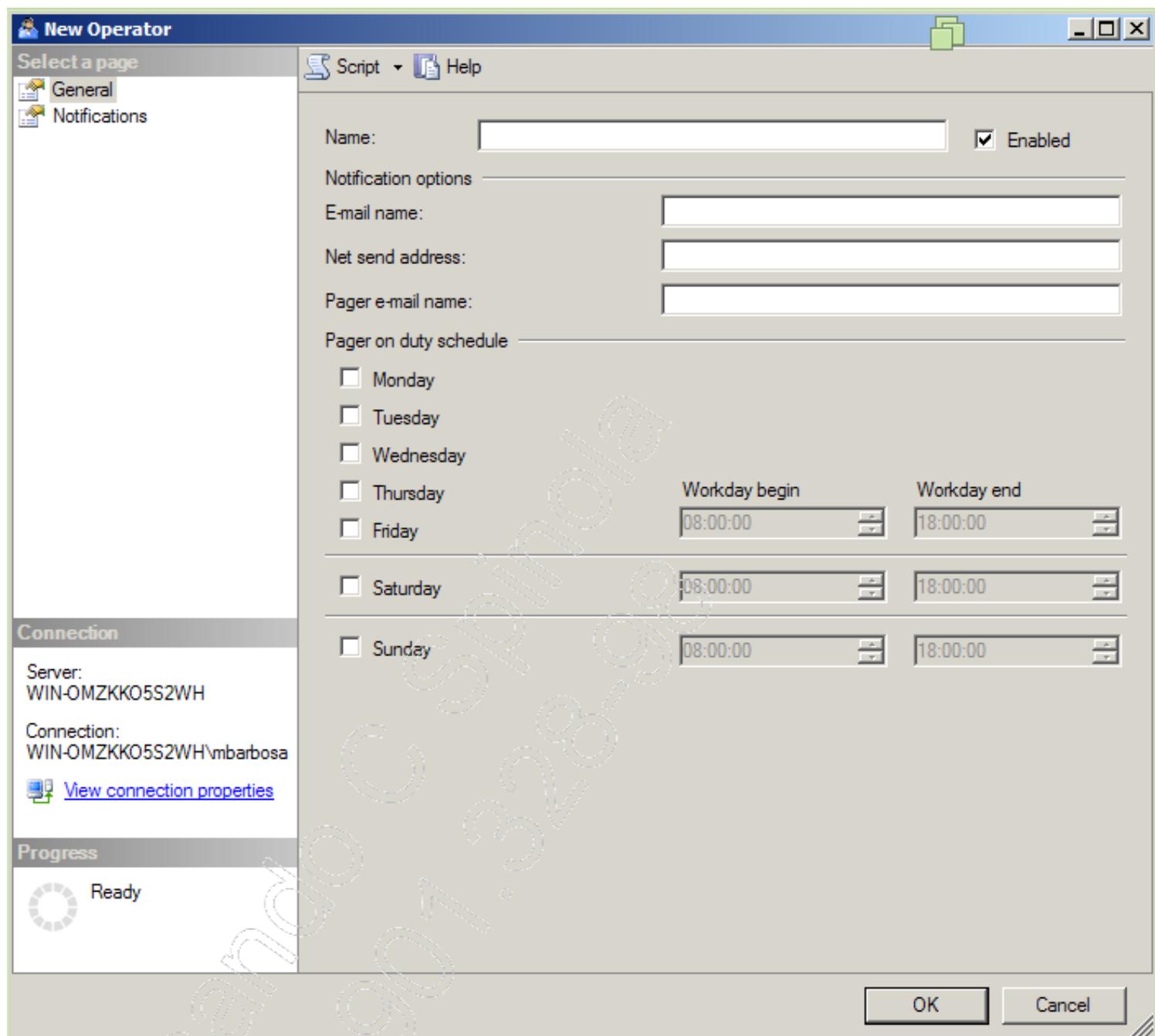
A seguir, vemos como criar operadores no SSMS. Primeiramente, selecione a opção **Operators** e clique no botão **New Operator...**.



Informe o nome do operador (**Name**) e as formas de notificação:

- E-mail (**E-mail name**);
- Endereço de envio de mensagem (**Net send address**);
- Endereço de pager (**Pager e-mail name**);
- Indique se o operador está habilitado ou não (**Enabled = Habilitado**).

SQL 2014 - Módulo III



Para criarmos um operador usando comandos, a sintaxe é a seguinte:

```
sp_add_operator [ @name = ] 'name'  
    [ , [ @enabled = ] enabled ]  
    [ , [ @email_address = ] 'email_address' ]  
    [ , [ @pager_address = ] 'pager_address' ]  
    [ , [ @weekday_pager_start_time = ] weekday_pager_start_time  
]  
    [ , [ @weekday_pager_end_time = ] weekday_pager_end_time ]  
    [ , [ @saturday_pager_start_time = ] saturday_pager_start_  
time ]  
    [ , [ @saturday_pager_end_time = ] saturday_pager_end_time ]  
    [ , [ @sunday_pager_start_time = ] sunday_pager_start_time ]  
    [ , [ @sunday_pager_end_time = ] sunday_pager_end_time ]  
    [ , [ @pager_days = ] pager_days ]  
    [ , [ @netsend_address = ] 'netsend_address' ]  
    [ , [ @category_name = ] 'category' ]
```

Em que:

- [**@name=**] ‘name’: Trata-se do nome de um operador (destinatário da notificação), que deve ser exclusivo. O caractere de porcentagem (%) não deve fazer parte desse nome;
- [**@enabled=**] **enabled**: Status do operador. Enabled tem o valor padrão definido como 1, indicando que está habilitado. O operador não estará habilitado e não receberá notificações quando o valor for 0 (zero);
- [**@email_address=**] ‘email_address’: Trata-se do endereço de e-mail do operador. Deve conter apenas um e-mail;
- **Parâmetros pager**: Foram mantidos por compatibilidade das versões anteriores;
- **[@Netsend_address=] ‘netsend_address’**: Endereço do usuário na rede Windows. Este parâmetro fará com que uma mensagem seja enviada via terminal Windows. A opção padrão é NULL.

Para eliminarmos um operador, a sintaxe do comando é a seguinte:

```
sp_delete_operator [ @name = ] 'name'  
[ , [ @reassign_to_operator = ] 'reassign_operator' ]
```

Em que:

- **[@name=] ‘name’**: O nome de um operador (destinatário da notificação). Esse nome deve ser exclusivo e não suporta expressões do tipo like (%);
- **[@reassign_to_operator =] ‘reassign_operator’**: O endereço de e-mail do operador para o qual os e-mails deverão ser reenviados.

Para obtermos informações de um operador, a sintaxe do comando é a seguinte:

```
sp_help_operator  
{ [ @operator_name = ] 'operator_name'  
| [ @operator_id = ] operator_id }
```

Em que:

- **[@operator_name=] ‘operator_name’**: Trata-se do nome de um operador (destinatário da notificação), o qual deve ser exclusivo. O caractere de porcentagem (%) não pode fazer parte desse nome.

Para alterarmos um operador, a sintaxe do comando é a seguinte:

```
sp_update_operator
[ @name = ] 'name'
[ , [ @new_name = ] 'new_name' ]
[ , [ @enabled = ] enabled]
[ , [ @email_address = ] 'email_address' ]
[ , [ @pager_address = ] 'pager_number' ]
[ , [ @weekday_pager_start_time = ] weekday_pager_start_time
]
[ , [ @weekday_pager_end_time = ] weekday_pager_end_time ]
[ , [ @saturday_pager_start_time = ] saturday_pager_start_
time ]
[ , [ @saturday_pager_end_time = ] saturday_pager_end_time ]
[ , [ @sunday_pager_start_time = ] sunday_pager_start_time ]
[ , [ @sunday_pager_end_time = ] sunday_pager_end_time ]
[ , [ @pager_days = ] pager_days ]
[ , [ @netsend_address = ] 'netsend_address' ]
[ , [ @category_name = ] 'category'
```

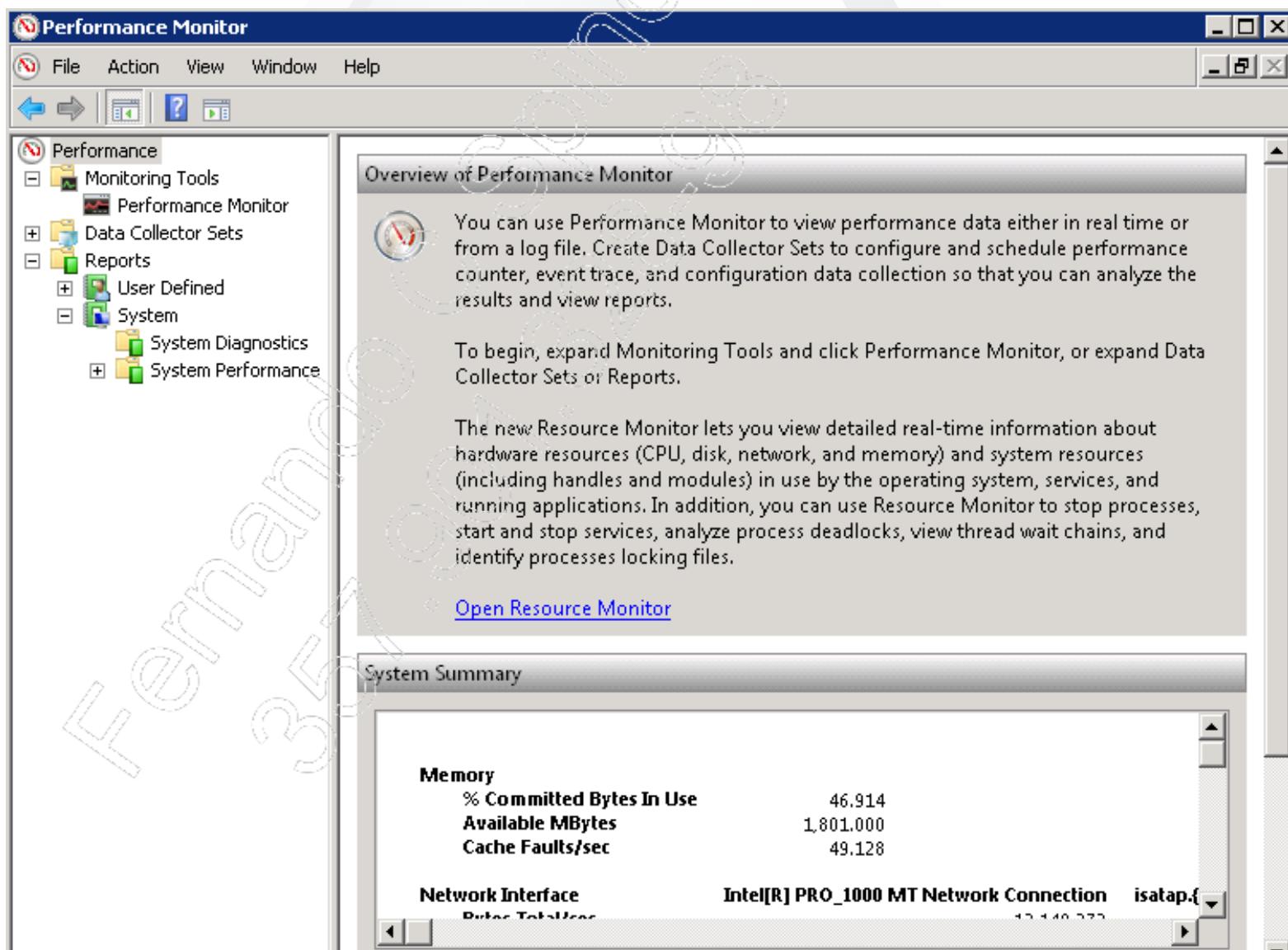
Em que:

- [**@name=**] ‘name’: O nome de um operador (destinatário da notificação). Esse nome deve ser exclusivo e não suporta expressões do tipo like (%);
- [**@enabled=**] enabled: Status atual do operador. Enabled tem o valor padrão definido como 1, indicando que está habilitado. O operador não estará habilitado e não receberá notificações quando o valor for 0 (zero);
- [**@email_address=**] ‘email_address’: Indica o endereço de e-mail do operador. Deve conter apenas um e-mail.

8.6. Configurando alertas (Alerts)

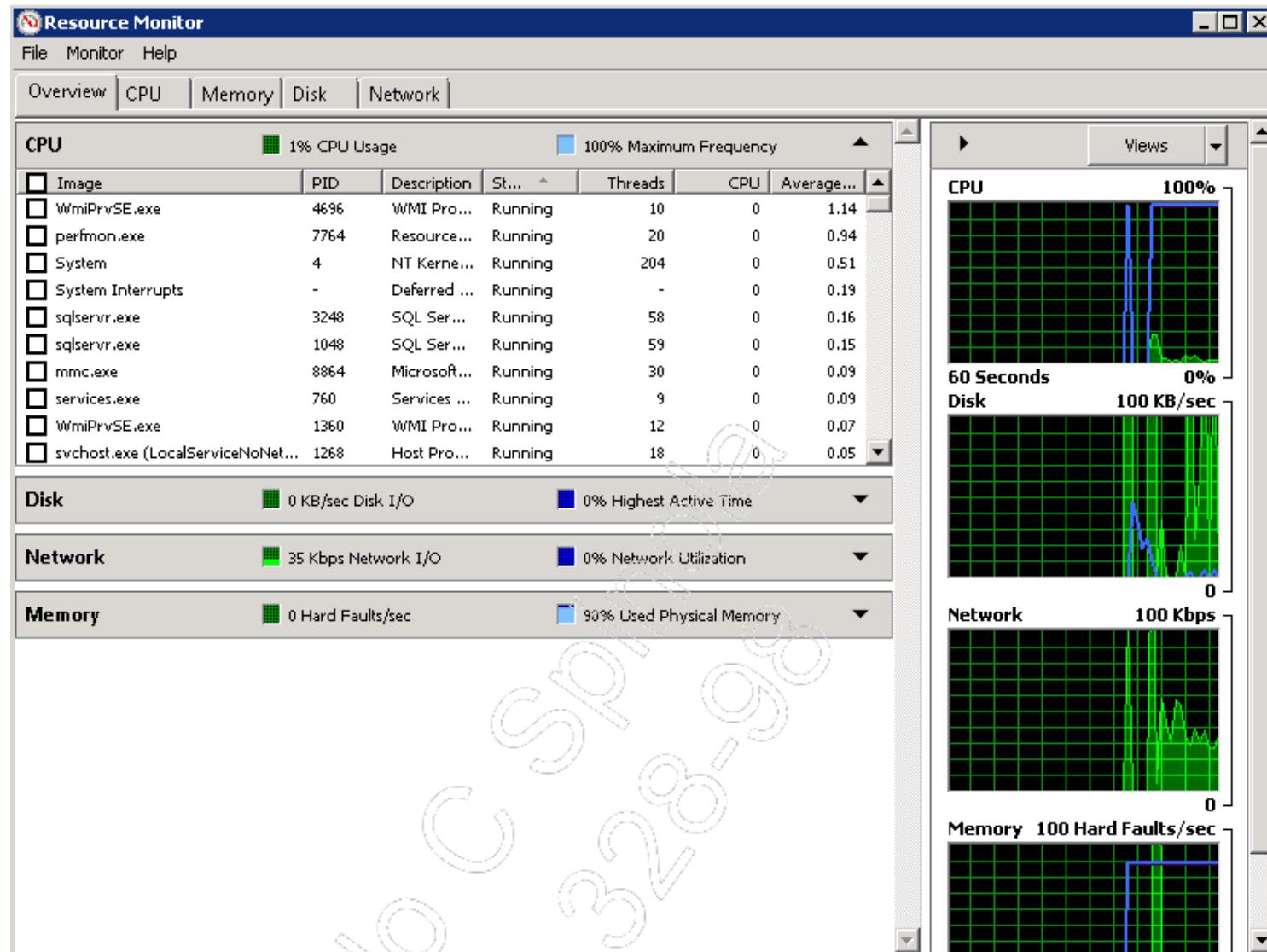
Alertas para o SQL Server funcionam com avisos, que configuramos quando certas atividades ocorrem. Eles podem estar vinculados a condições ligadas à performance do banco de dados. Quando estas condições ocorrem, podemos vincular alertas a elas, conforme os valores apresentados.

Os alertas podem estar vinculados a contadores existentes em uma ferramenta do Windows chamada **Performance Monitor**, que registra elementos para monitoração do sistema operacional. Ao instalarmos o SQL Server, são criados novos pontos de monitoração nesta ferramenta. Para acessar esta ferramenta, vá até o botão **Start**, escolha a opção **Run** e digite **PERFMON.EXE**. A tela a seguir será exibida:



Automação de tarefas, alertas e operadores

Selecione o link **Open Resource Monitor** e a tela a seguir será exibida:

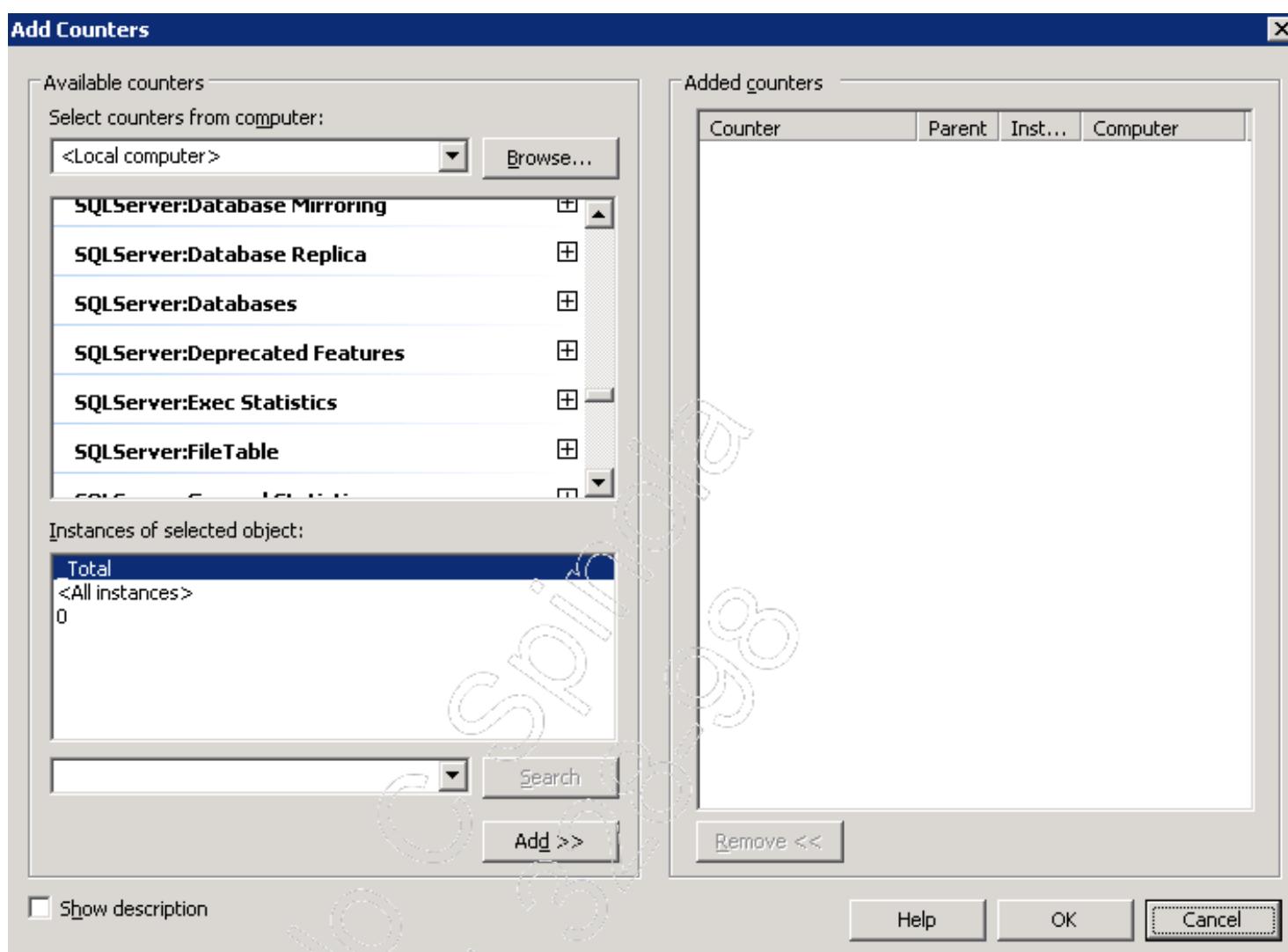


Por meio dela, podemos obter um grande conjunto de informações a respeito da máquina e do sistema operacional, sem contar vários indicadores do SQL Server.

Selecione a opção de desempenho de sistema (**System Performance**) e clique no botão de adição (+) na barra de ferramentas.

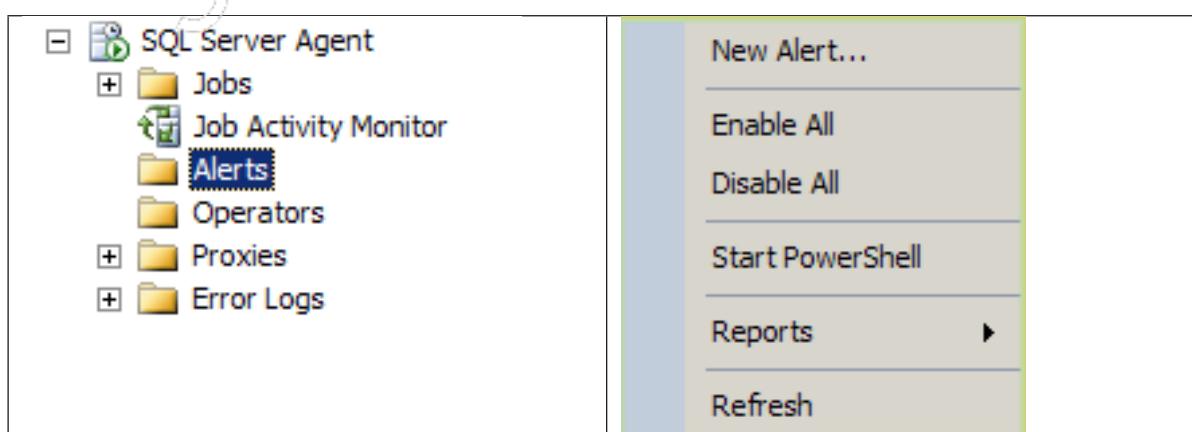
SQL 2014 - Módulo III

A tela **Add Counters** a seguir será exibida e vários contadores poderão ser adicionados através da ferramenta **Performance Monitor**.



Além disso, os alertas criados dentro do SQL Server estão relacionados aos mesmos contadores (counters) vistos na ferramenta **Perfmon.exe**.

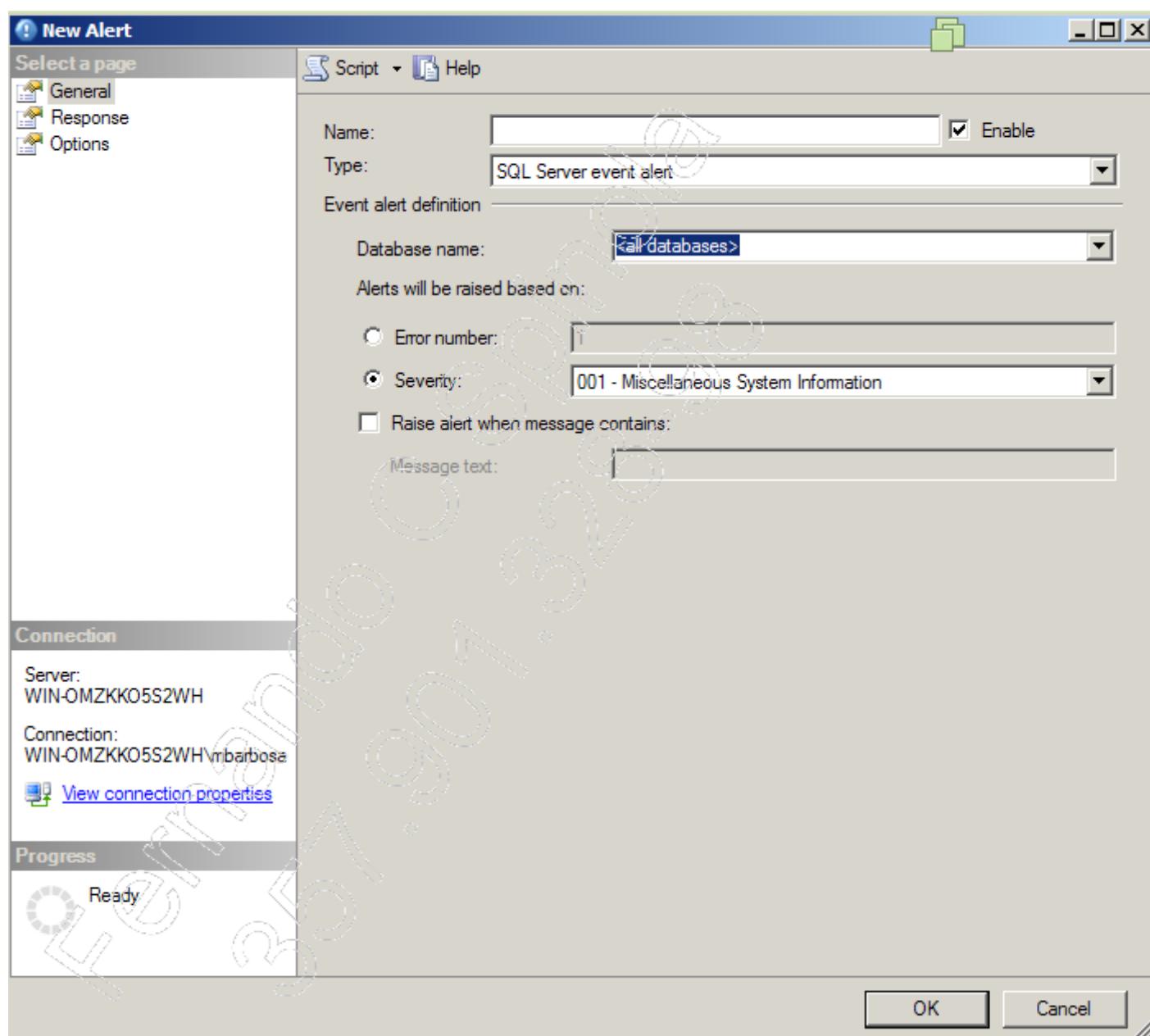
Vejamos a seguir como criar alertas usando o SSMS. Primeiramente, selecione a opção **Alerts** no SQL Server Agent e clique na opção **New Alert....**



Automação de tarefas, alertas e operadores

Vamos definir o nome dado ao alerta (**Name**). Em seguida, precisamos indicar que esse alerta está habilitado (**Enable**) e escolher um destes tipos de alerta:

- SQL Server event alert;
- SQL Server performance condition alert;
- WMI event alert (**Perfmon**).

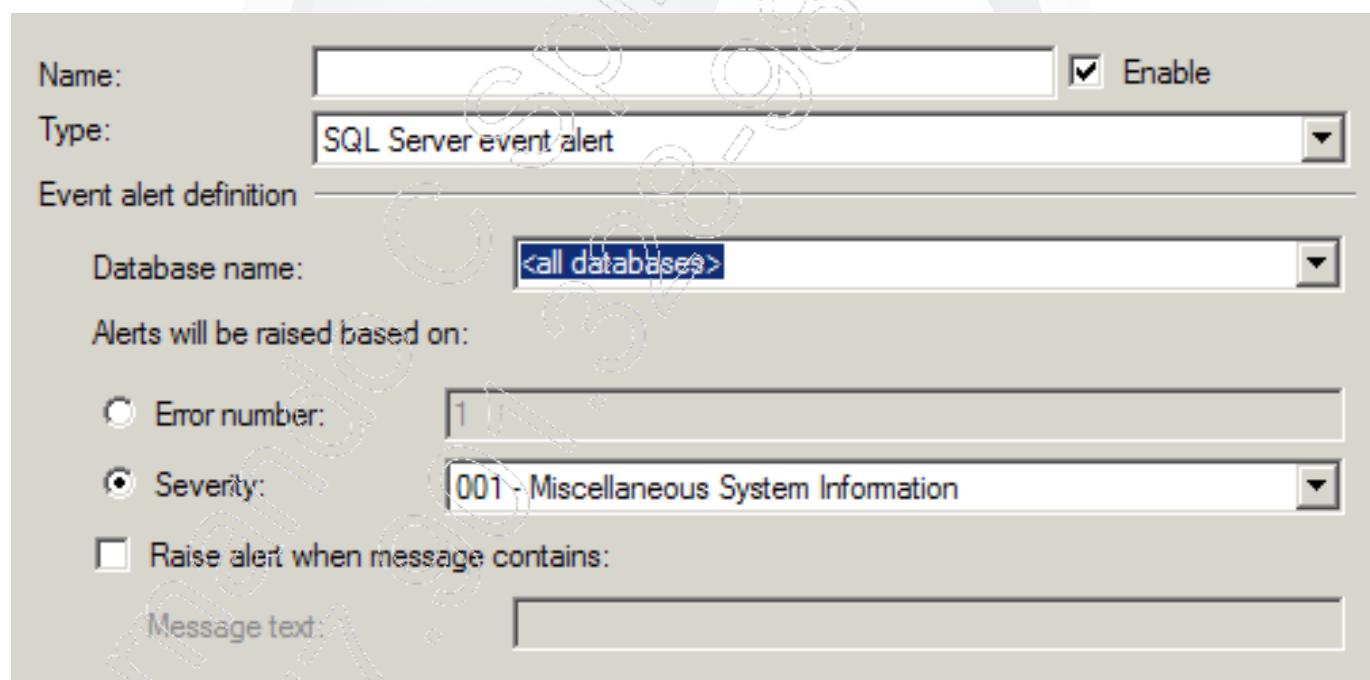


- Caso a opção seja **SQL Server event alert**

Devemos indicar o nome do banco de dados cujo alerta desejamos que esteja ligado (**Database alert**). Além disso, precisamos definir quando os alertas deverão ser invocados:

- Número do erro (**Error number**);
- Severidade do erro (**Severity**).

Opcionalmente, podemos indicar que o alerta pode ser invocado quando um determinado texto for encontrado na mensagem de erro.



Os códigos e mensagens de erros podem ser obtidos na visão **sys.messages**:

```
SELECT * FROM SYS.SYSDATABASES
```

Podemos também criar novas mensagens de erros para uso de aplicações (gatilhos, procedimentos etc.). Esses erros ocorrerão apenas mediante a invocação explícita em uma aplicação.

Para criar uma nova mensagem, o comando é:

```
SP_ADD_MESSAGE @error_num, @severity, @message, @language, @save
```

Em que:

- **Error_num**: Número do erro. Deve ser superior a 50000;
- **Severity**: Severidade do erro 1 a 25;
- **Message**: Mensagem de erro desejada. Em caso de mensagens de erro parametrizadas, podemos usar os literais %d para parâmetros numéricos e %s para caracteres. Ao chamar erros com parâmetros, o programa que invocou o erro precisa informar os valores;
- **Save**: Indica se deseja gravar o erro no log de aplicativos do Windows.

Vejamos, a seguir, o exemplo para criação de mensagem:

```
EXEC SP_ADDMESSAGE 50001, 16, 'ERRO NA APLICACAO', 'us_english'  
, 'true'
```

SQL 2014 - Módulo III

- Caso a opção seja SQL Server performance condition alert

Devemos indicar o objeto do SQL Server que se deseja monitorar:

Access Methods
Availability Replica
Batch Resp Statistics
Broker Activation
Broker Statistics
Broker TO Statistics
Broker/DBM Transport
Buffer Manager
Buffer Node
Catalog Metadata
CLR
Cursor Manager by Type
Cursor Manager Total
Database Replica
Databases
Deprecated Features
Exec Statistics
FileTable
General Statistics
Latches
Locks
Memory Broker Clerks
Memory Manager
Memory Node
Plan Cache
Query Execution
Resource Pool Stats
SQL Errors
SQL Statistics
Transactions

Segundo o objeto selecionado, as opções para contadores (counters) é modificada. Informe o tipo de contador adequado. Em seguida, selecione o valor desejado para o componente da instância que se deseja monitorar:

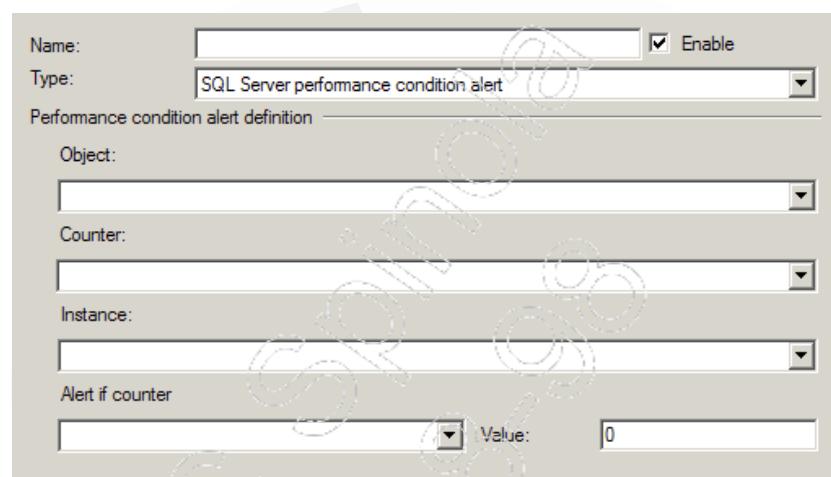
Total
AllocUnit
Application
Database
Extent
File
HoBT
Key
Metadata
Object
OibTrackTbl
Page
RID

Automação de tarefas, alertas e operadores

Indique, por fim, qual é o tipo de contagem que deverá ser usada (**Alert if counter**):

- Estiver abaixo de (**Falls below**);
- Tornar-se igual a (**Becomes equal to**);
- Estiver acima de (**Raises above**).

Indique o valor correspondente à opção anterior no campo valor (**Value**).



- Caso a opção seja WMI event alert

Devemos, neste caso, digitar a consulta que desejamos realizar quanto ao alerta. Vejamos um exemplo a seguir:

```
SELECT * FROM DEADLOCK_GRAPH
```

SQL 2014 - Módulo III

Vejamos, a seguir, um exemplo de uso para **WMI Alert**:

```
USE PEDIDOS ;
GO

IF OBJECT_ID('EventoDeadlock', 'U') IS NOT NULL
BEGIN
    DROP TABLE EventoDeadlock ;
END ;
GO

CREATE TABLE EventoDeadlock
    (DataAlerta DATETIME, GraficoDeadlock XML) ;
GO
-- Adiciona uma tarefa

EXEC msdb.dbo.sp_add_job @job_name=N'CAPTURA_DEADLOCK',
    @enabled=1,
    @description=N'Tarefa associada a DEADLOCK_GRAPH' ;
GO

-- Adiciona um passo que insere o horário atual e o gráfico de
deadlock
-- na tabela EventoDeadlock

EXEC msdb.dbo.sp_add_jobstep
    @job_name = N'CAPTURA_DEADLOCK',
    @step_name=N'Insert graph into LogEvents',
    @step_id=1,
    @on_success_action=1,
    @on_fail_action=2,
    @subsystem=N'TSQL',
    @command= N'INSERT INTO EventoDeadlock
        (DataAlerta, GraficoDeadlock)
        VALUES (getdate(), N'''$(ESCAPE_
SQUOTE(WMI(TextData)))''''),
    @database_name=N'PEDIDOS' ;
GO

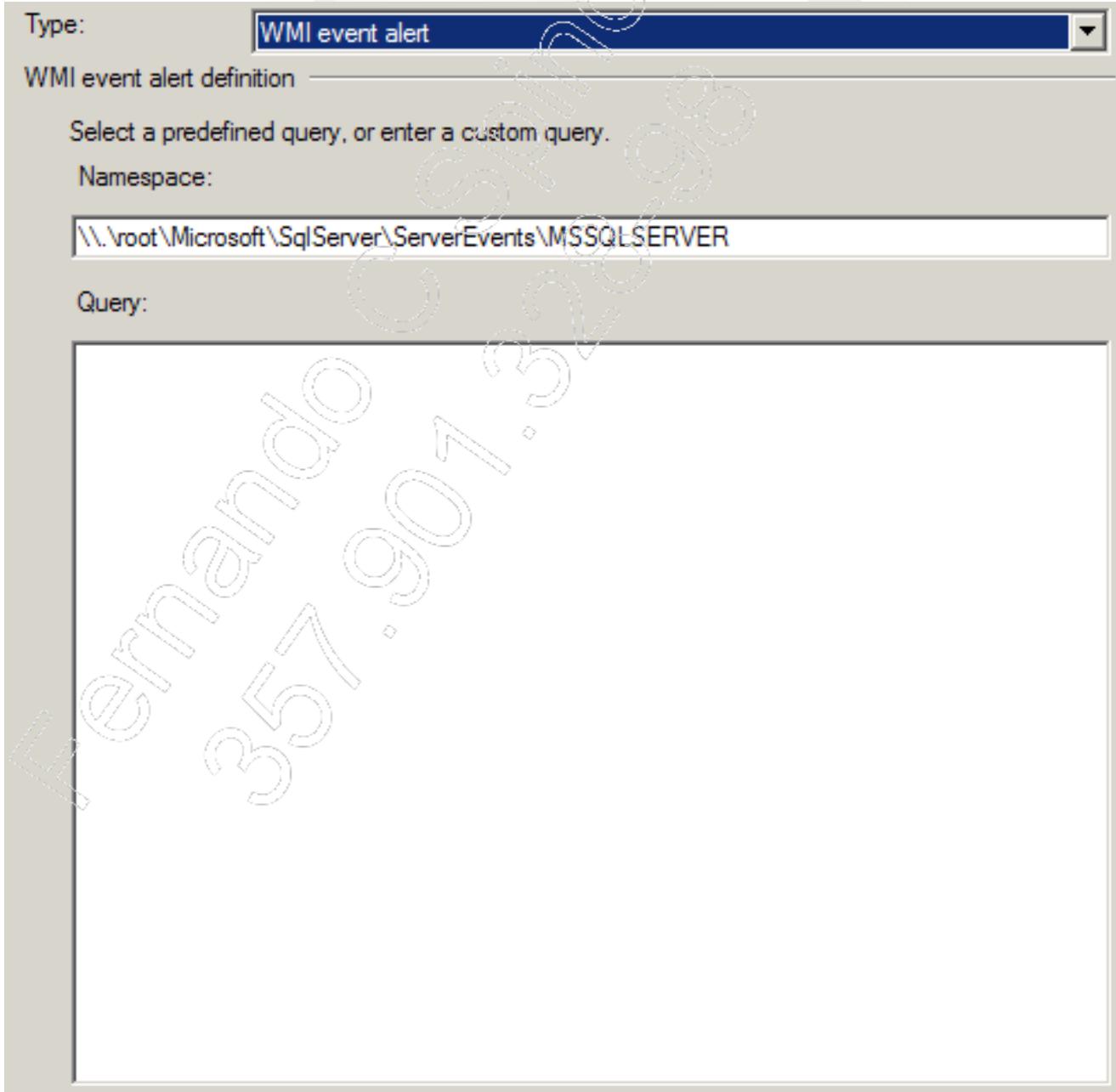
-- Indica que a instância corrente será associada à tarefa.
```

Automação de tarefas, alertas e operadores

```
EXEC msdb.dbo.sp_add_jobserver @job_name = N'CAPTURA_DEADLOCK' ;
GO

-- Adicione um alerta que responda a todos os eventos DEADLOCK_
GRAPH para a
-- instância default

EXEC msdb.dbo.sp_add_alert @name=N'GRAFICO DEADLOCK',
@wmi_namespace=N'\\.\root\Microsoft\SqlServer\ServerEvents\
MSSQLSERVER',
@wmi_query=N'SELECT * FROM DEADLOCK_GRAPH',
@job_name='CAPTURA_DEADLOCK' ;
GO
```

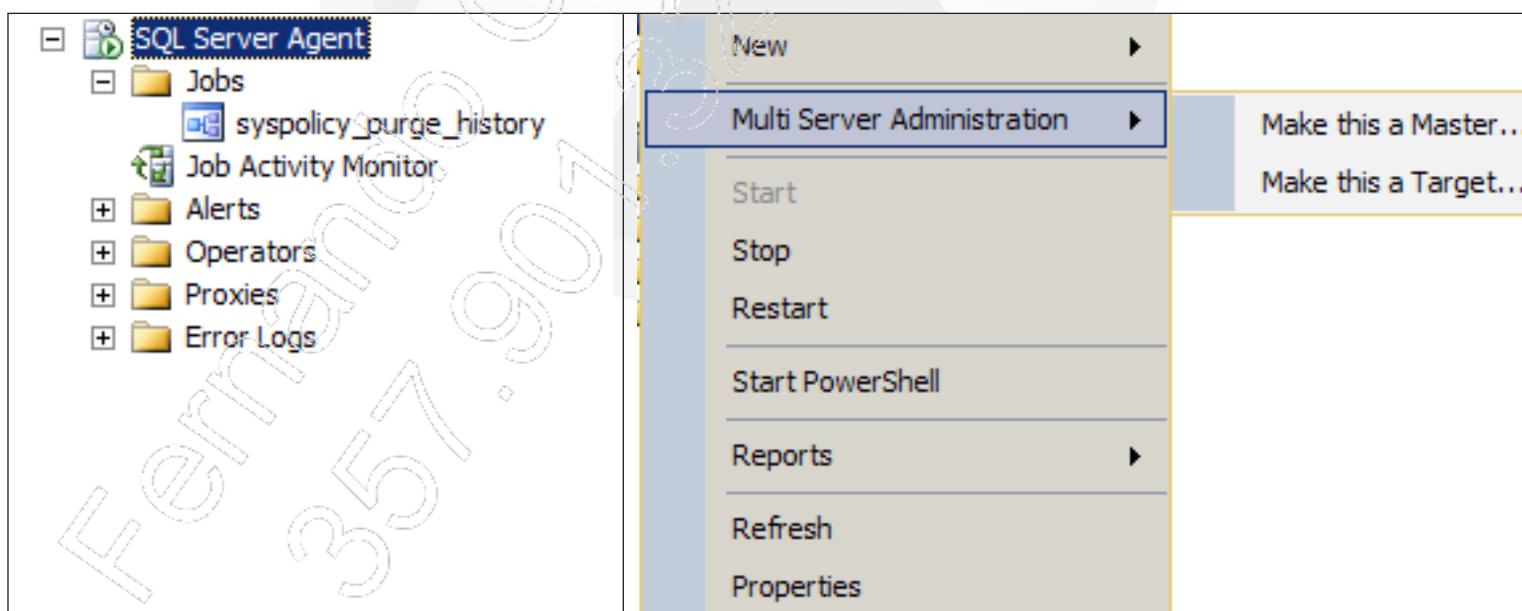


8.7. Configurando múltiplos SQL Server Agents utilizando centralização

À medida que a complexidade e o número de servidores SQL Server aumentam, pode ser necessária uma centralização de controle dos SQL Server Agents para que a administração de tarefas e o gerenciamento de alertas e operadores sejam mais eficientes.

Nesse sentido, devemos estruturar e organizar os servidores para que a operação seja feita em conjunto, por exemplo, organizando grupos de servidores de produção, homologação e testes.

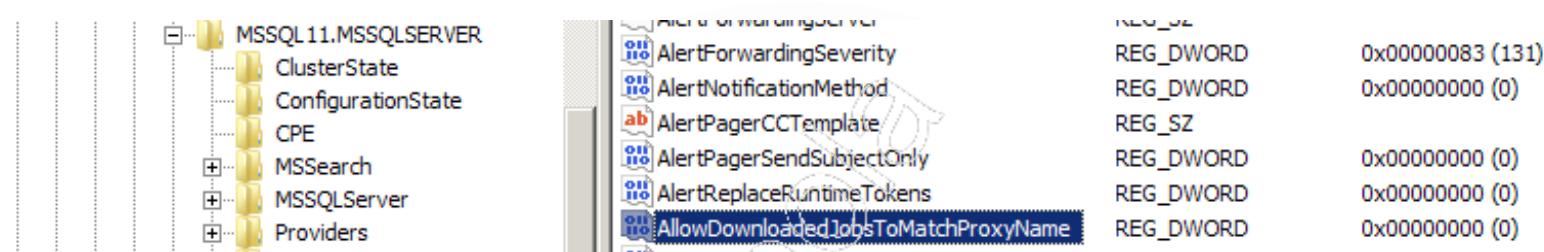
A gestão de múltiplos ambientes com centralização precisa estabelecer uma instância que será a centralizadora das atividades do SQL Server Agent. Essa instância será o agente Master e, a partir dele, as tarefas (**Jobs**) serão agendadas nesse servidor para serem executadas nos demais servidores.



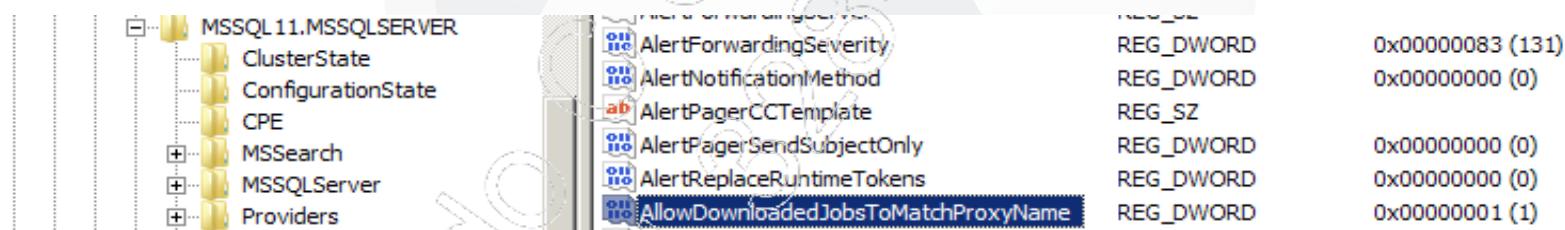
Automação de tarefas, alertas e operadores

Antes de iniciar, precisamos alterar a entrada do regedit, que trata desta questão. Para isso, clique no botão **Start**, selecione **Run** e digite **REGEDIT.EXE**.

Localize no regedit a seguinte pasta: **\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\<instance_name>\SQL Server Agent\AllowDownloadedJobsToMatchProxyName**.

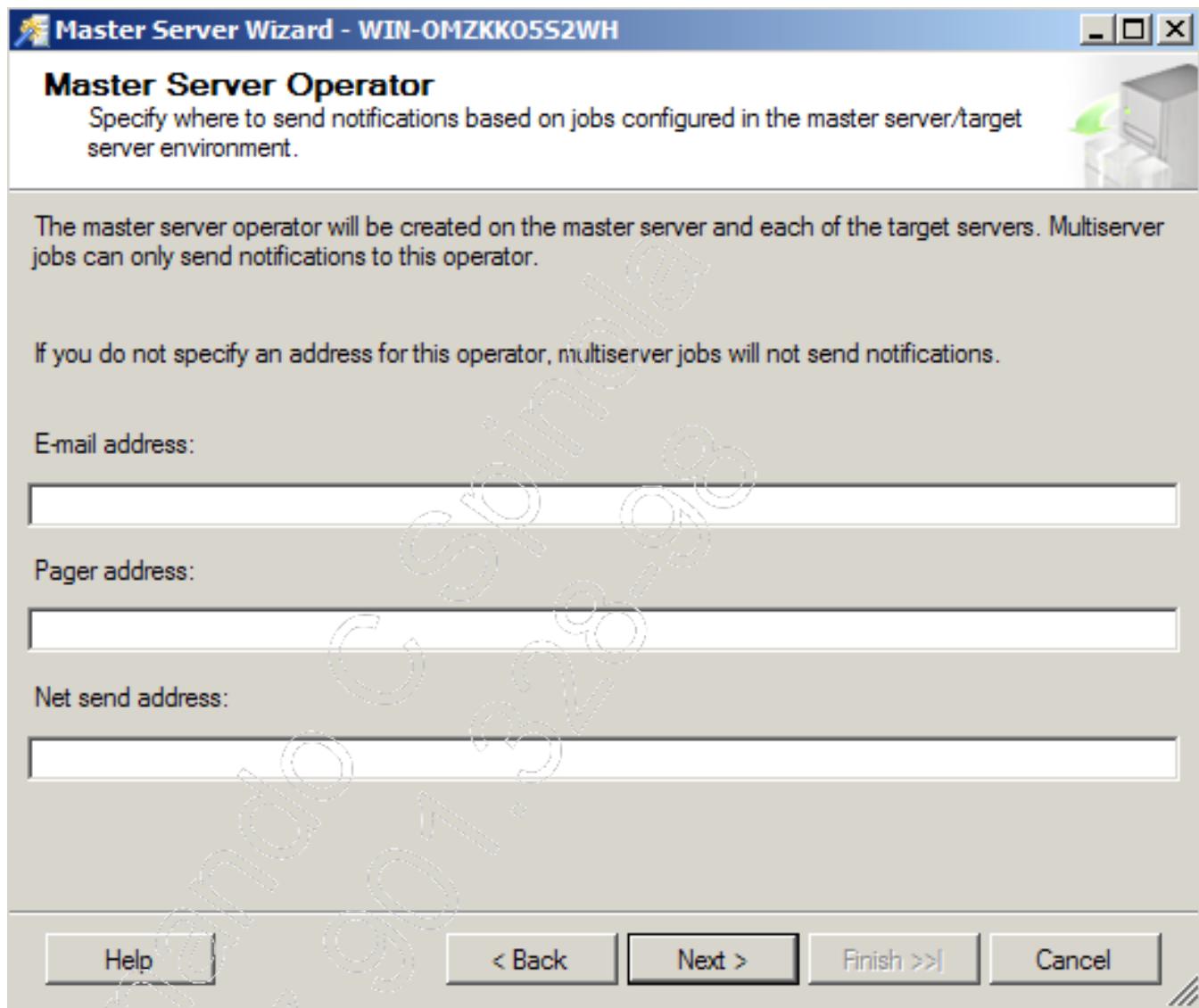


Altere o valor de 0 para 1.



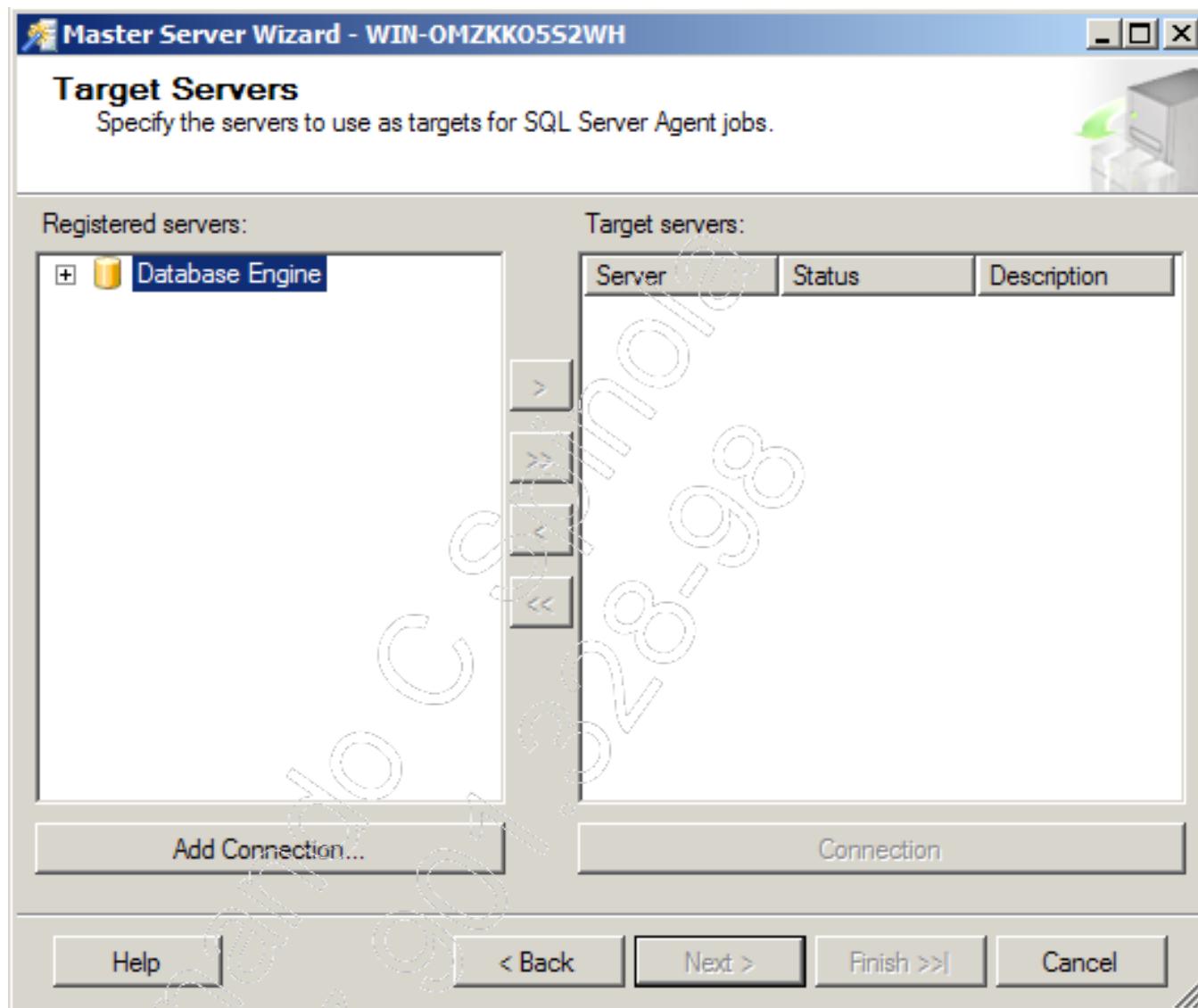
É necessário transformar um dos SQL Server Agents em Master, usando a opção **Make this a Master....**

Informe o endereço de e-mail (**E-mail address**) e/ou endereço do pager (**Pager address**) e/ou o endereço de mensagem de rede (**Net send address**).



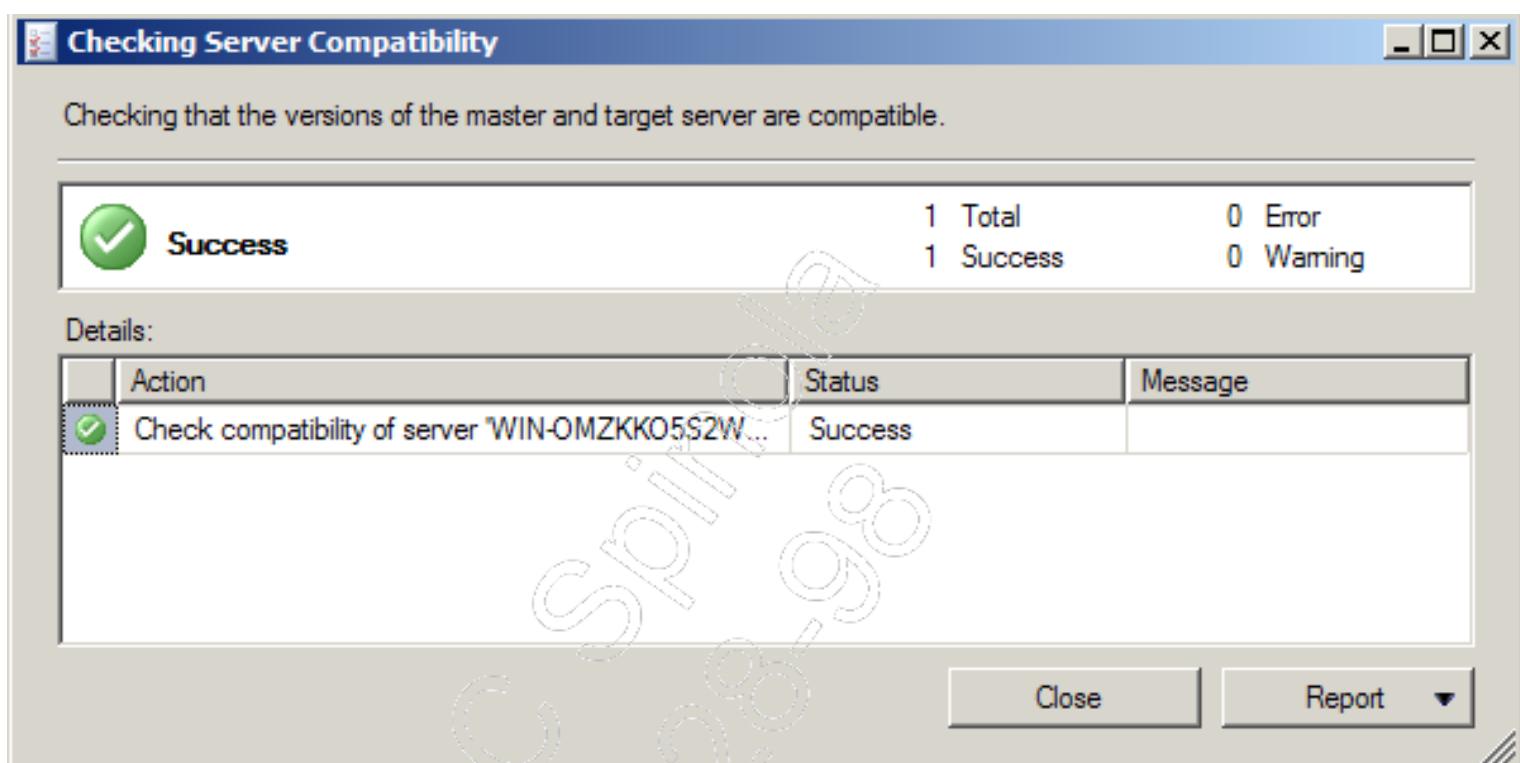
Selecione o servidor na opção **Registered servers**. Escolha a opção **Local Server Group** e clique no servidor da opção.

Em seguida, clique no botão > para adicionar esse servidor a um servidor de destino (**Target servers**). Em seguida, clique no botão **Next**.

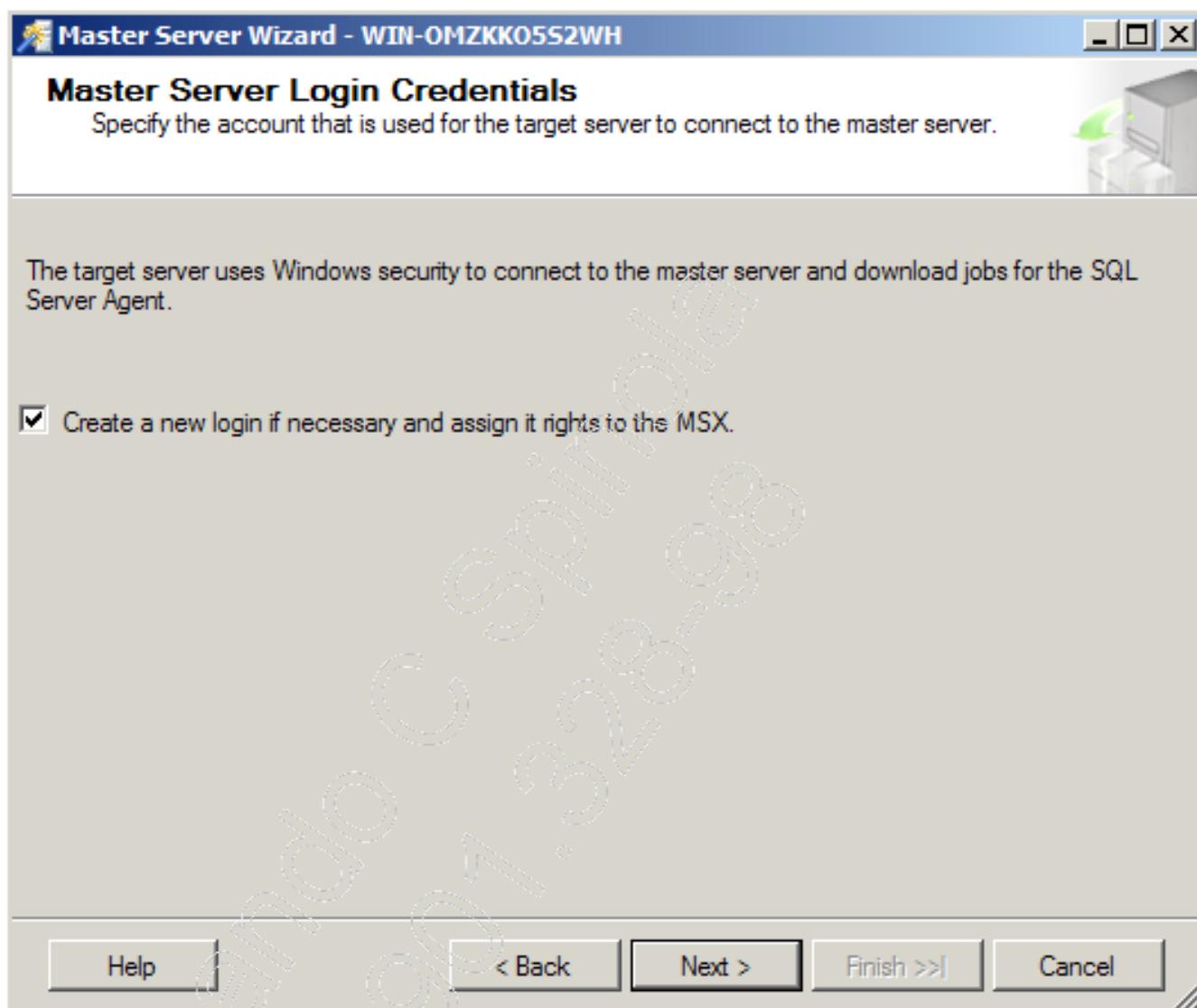


SQL 2014 - Módulo III

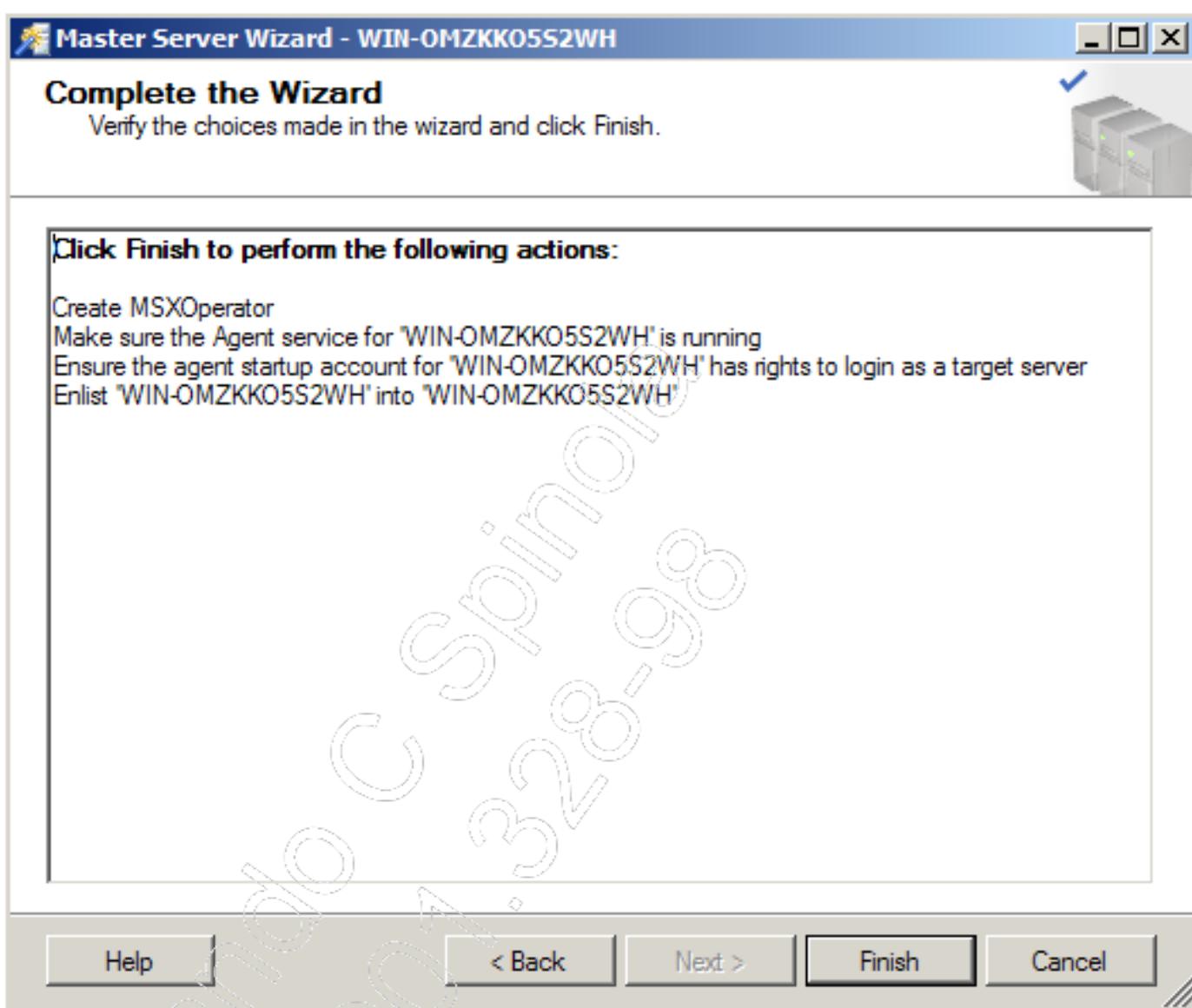
Caso a compatibilidade de servidor seja bem-sucedida (**Success**), clique no botão fechar (**Close**).



Nesta tela, será criada a credencial de login com os direitos de acesso necessários.



Clique na opção **Finish** para encerrar o processo de configuração.

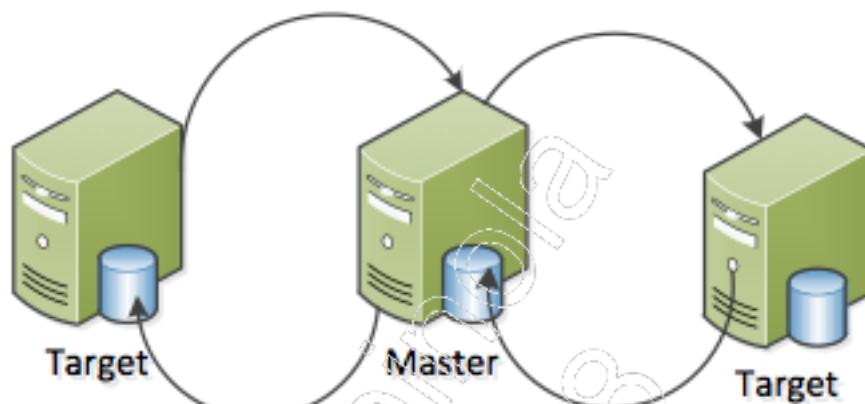


Nos servidores master, é possível criar tarefas (**Jobs**) que são executadas nos servidores alvo/destino (**target**). Essas tarefas são registradas na tabela **sysdownloadlist**, que pertence ao banco de dados MSDB. Periodicamente, esses servidores se conectam ao servidor master para verificar se existe alguma tarefa para ser executada. Caso isso ocorra, o servidor alvo/destino carrega a tarefa e a desconecta do servidor master. Ao terminar a tarefa, o servidor alvo/destino se reconecta ao master a fim de atualizar o status da execução da referida tarefa.

A seguir, vejamos como é feita a consulta à tabela que registra as tarefas carregadas pelos servidores alvo/destino e suas execuções:

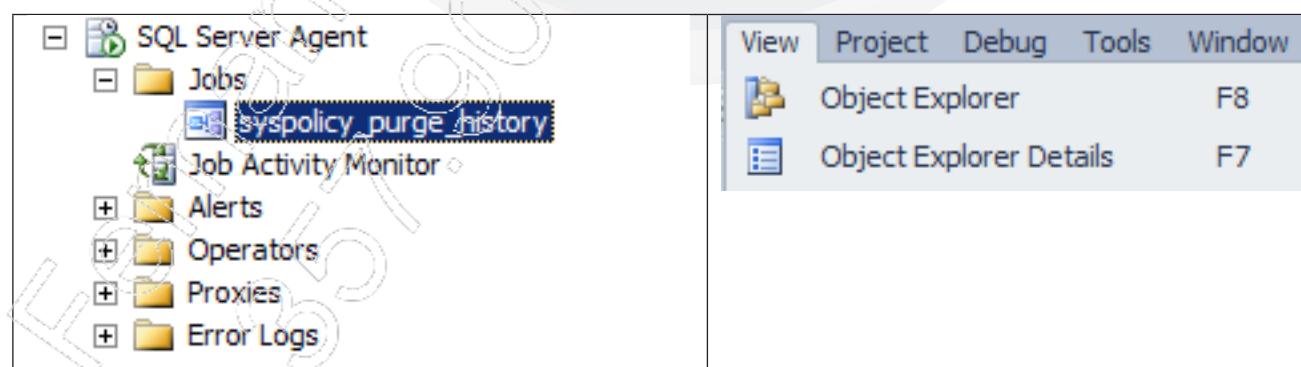
```
Select * from msdb.sysdownloadlist
```

Vejamos um esquema da conexão entre master e dois servidores alvo/destino:



8.8. Fazendo cópia de todas as tarefas

Muitas vezes, quando a administração de tarefas não é centralizada, é necessário passar um ou mais Jobs entre duas ou mais instâncias. A forma mais simples de fazer essa operação é a descrita a seguir:



SQL 2014 - Módulo III

Pressione F7 para abrir a janela de detalhes. Em seguida, selecione a tarefa:

The screenshot shows the 'Object Explorer Details' window in SQL Server Management Studio. The title bar indicates two open queries: 'SQLQuery5.sql' and 'SQLQuery2.sql'. The main pane displays the 'Jobs' node under the 'WIN-OMZKKO5S2WH' server. A single job, 'syspolicy_purge_history', is listed in the table, showing it is 'Not running'.

Pressione o botão esquerdo do mouse sobre essa tarefa e selecione a opção **CREATE To**:

The screenshot shows the context menu for the 'syspolicy_purge_history' job. The 'CREATE To' option is highlighted with a yellow box. Other options visible in the menu include 'New Job...', 'Start Job at Step...', 'Stop Job', 'Script Job as', 'View History', 'Enable', and 'Disable'.

Isso fará com que o Job fique todo sob o formato de script e, com isso, possa ser migrado para outro servidor.

 Cuidado, pois a mesma operação deverá ser feita com operadores (**Operators**) e alertas (**Alerts**), os quais devem ser criados antes das tarefas.

8.9. Solução de problemas (Troubleshooting)

Tendo em vista que problemas com tarefas (**Jobs**), alertas (**Alerts**) e operadores (**Operators**) podem ocorrer, devemos ter em mente algumas possíveis soluções para as situações de erros mais conhecidas. Para isso, vamos tratar de alguns fatores:

- **SQL Server Agent ligado**

A não execução de uma tarefa pode ser decorrente de uma falha nessa ação, ou pode ser que o serviço do SQL Server Agent não esteja ativo. Verifique se o serviço está ativo usando o gerenciador de serviços do Windows ou a ferramenta de gerenciamento de serviços do SQL Server. Caso esteja on-line, verifique o histórico de execução da tarefa.

- **Direitos administrativos da conta do login do SQL Server Agent**

O bom funcionamento do SQL Server Agent está ligado ao usuário que esse serviço utiliza para a inicialização no serviço do Windows. Essa conta também deverá estar mapeada para o privilégio da role **SYSADMIN**.

- **Permissões do proprietário da tarefa**

É muito importante que o proprietário da tarefa tenha permissões para realizá-la. Isso minimiza erros de execução que possam ocorrer.

- **Funcionamento do Database Mail**

Verifique periodicamente se os serviços de Database Mail estão funcionando. Caso tenha problemas na instalação, contate o administrador Windows e veja se o servidor de banco de dados SQL Server possui relay SMTP para envio de e-mail através do servidor de e-mail da empresa.

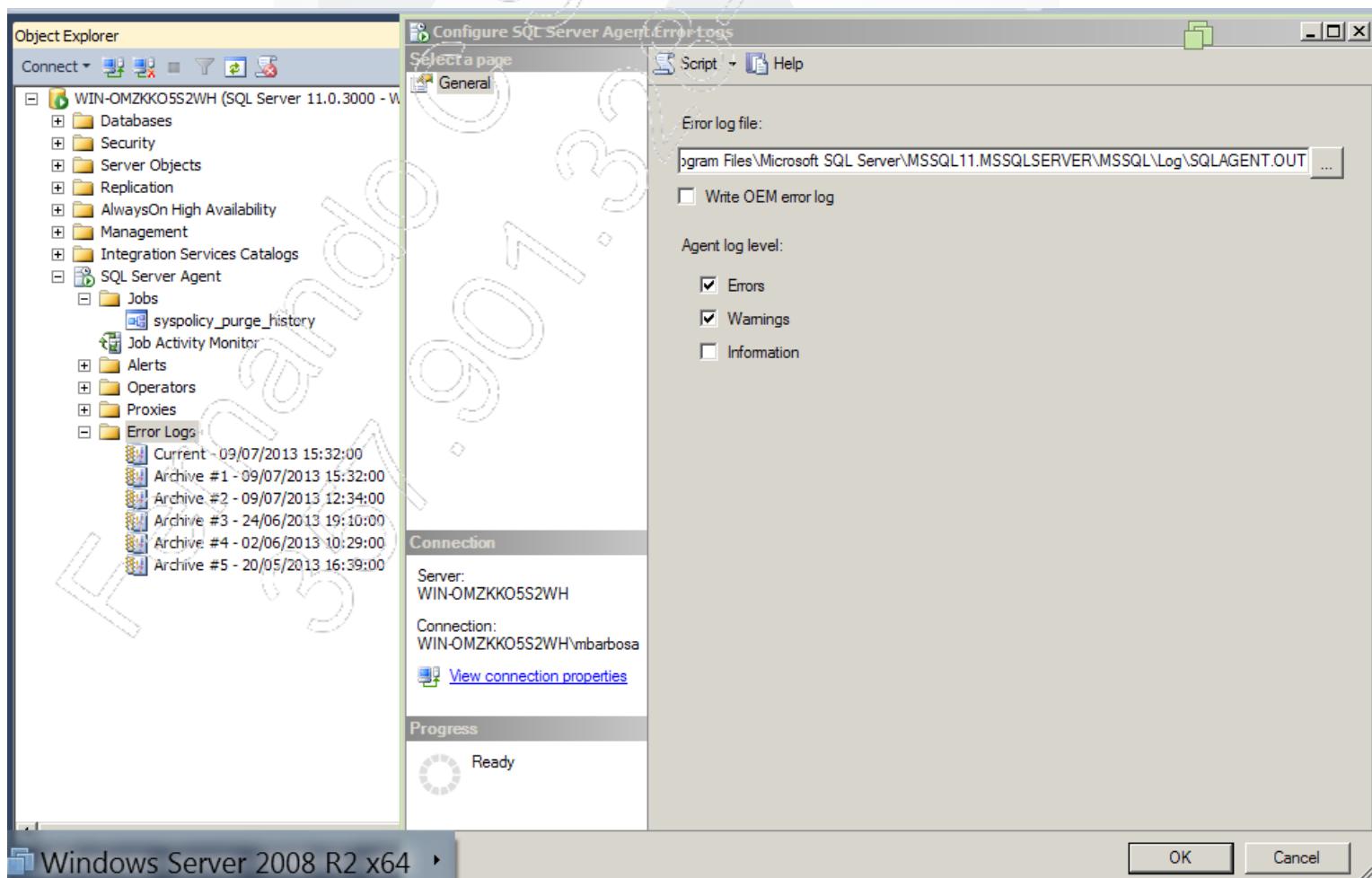
- **Tamanho do log de eventos de aplicativos do Windows**

Verifique regularmente o log de eventos do Windows (**Event Viewer**), pois este arquivo normalmente é sobreescrito e, portanto, as informações nele contidas se perdem. Caso muitos erros sejam gravados no log de eventos, é importante fazer cópias regulares desse arquivo.

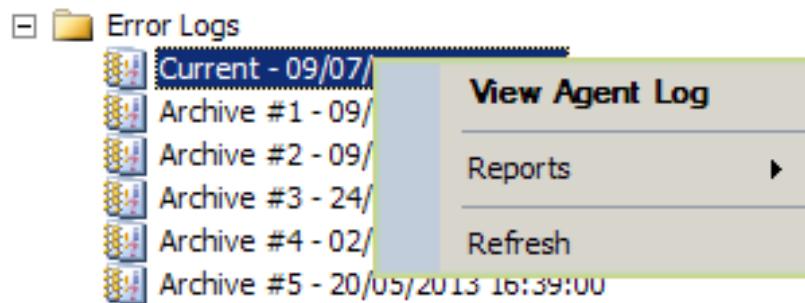
- **Histórico**

O histórico de tarefas, alertas e operadores fornecem informações para detectar possíveis erros ou falhas, por isso, devem ser verificados periodicamente. Essas informações são gravadas em arquivos de logs do SQL Server Agent (não confunda com arquivos de log do banco de dados).

Selecione a opção **Error Logs**, pressione o botão esquerdo do mouse sobre a opção **Configure** e veja o caminho dos arquivos de erros do log do SQL Server Agent. Você pode incluir ou retirar a opção de gravação de erros, avisos e informações.



Caso queira ver os registros deste arquivo, posicione o cursor do mouse sobre o arquivo desejado e aplique um duplo-clique. O arquivo será exibido. Você ainda pode optar por posicionar o cursor do mouse sobre o arquivo desejado, clicar com o botão esquerdo e selecionar **View Agent Log**, conforme mostra a imagem a seguir:



- **Atraso no envio do alerta**

O processamento do alerta pode sofrer atraso mesmo que o erro já tenha ocorrido. Isso é possível se o uso de CPU estiver muito elevado ou se houver movimento muito alto de gravação do log de aplicativo de eventos do Windows (**Event Viewer**).

- **Desabilitação de tarefas**

Tenha cuidado ao desabilitar uma tarefa, pois, quando for ativada, ela não irá executar todas as tarefas que deixou de fazer anteriormente.

Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo.

- A centralização de tarefas em um servidor Master, em um ambiente com múltiplos SQL Server Agents, é uma atividade que precisa ser tratada e organizada pelo administrador;
- É muito importante organizar as tarefas a serem executadas em horários previamente agendados;
- O SQL Server Agent é um componente executado através de serviços do Windows (**Services.msc**);
- O gerenciamento de performance do Windows em um servidor SQL Server contempla, além de monitoramento de recursos do Windows, o monitoramento de recursos do SQL Server (**Perfmon.exe**);
- O recurso **Database Mail** é recomendado para envio de e-mails para usuários, tanto na execução de tarefas quanto na execução de rotinas de programação (gatilhos, procedimentos etc.);
- A extended procedure **xp_sendmail** só deve ser usada até o SQL 2008, pois, a partir da versão 2012, ela foi descontinuada;
- Várias tarefas podem ser agendadas pelo administrador de banco de dados (DBA), entre elas: recriação de índices, execução de estatísticas, backup de banco de dados e de log, compactação e truncamento de log, verificação de integridade dos arquivos.

8

Automação de tarefas, alertas e operadores

Teste seus conhecimentos

Fernando Cipriano
357.907.



IMPACTA
EDITORA

1. Que recurso do Windows o SQL utiliza para o monitoramento de performance?

- a) Editor de Performance
- b) SQL Configure
- c) SQL Monitor
- d) PERFMON
- e) PROFILE

2. Qual afirmação está errada sobre Jobs?

- a) É necessário definir um proprietário.
- b) Podemos criar agendas para a execução.
- c) Suporta comandos: T-SQL, Powershell, Activex e batch do SO.
- d) São lentos e não devem ser utilizados.
- e) Ao criá-los, as tarefas ficam habilitadas.

3. O que são operadores do SQL Server?

- a) São usuários que têm a finalidade de receber notificações e e-mails.
- b) São usuários.
- c) São usuários da ROLE SYSADMIN.
- d) São logins associados aos usuários.
- e) São usuários da ROLE SYSADMIN e DB_OWNER.

4. Qual afirmação está errada sobre envio de e-mail do SQL Server?

- a) Não deve ser utilizado, pois é um recurso antigo.
- b) O SQL Server utiliza o Database mail para envio de e-mail.
- c) SP_SEND_DBMAIL e XP_SENDMAIL são utilizados até a versão 2008.
- d) Pode ser utilizado para envio de mensagens de execução de tarefas.
- e) SQL Mail está em desuso desde a versão 2008.

5. O que ocorre quando o SQL Server Agent não está ativo?

- a) O SQL Server continua executando os jobs, mas não envia e-mail.
- b) O repositório para de funcionar.
- c) O SQL envia uma mensagem de erro, porém continua funcionando.
- d) O SQL para de executar o envio de e-mail.
- e) Tarefas agendadas não são executadas e nenhum e-mail é enviado.

Automação de tarefas, alertas e operadores

8

Mãos à obra!

Fernando
357.907.



IMPACTA
EDITORA

Observações iniciais

1 – Ao longo dos laboratórios deste capítulo, realizaremos as seguintes tarefas:

- Configuração de Operators;
- Configuração do Database Mail;
- Configuração de Jobs;
- Configuração de Alerts.

2 – Scripts utilizados nestes laboratórios

A pasta **Capítulo_08** contém todos os scripts com os comandos necessários para a realização destes laboratórios. Utilize esses scripts se não desejar digitar os respectivos comandos ou apenas para corrigir a sua execução.

3 – Forma de conexão no sistema operacional

Para realizar os laboratórios deste capítulo, conecte-se ao Windows com a conta de aluno cuja senha é **password**. Para tanto, utilize CTRL + ALT + DEL e escolha a opção **Logoff**. Em seguida, utilize CTRL + ALT + DEL novamente e conecte-se ao Windows com o login **Alunox** (em que x representa o seu número do aluno em sala de aula).

Laboratório 1

A – Configurando operadores

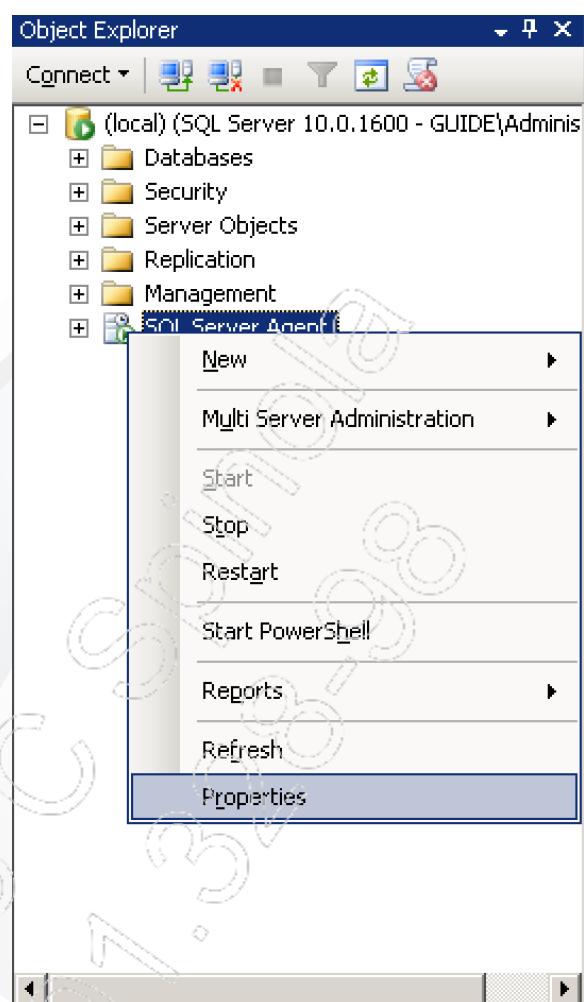
1. Do lado esquerdo da tela, no **Object Explorer** do **Microsoft SQL Server Management Studio**, expanda **SQL Server Agent**;
2. Expand a pasta **Operators**;
3. Clique com o botão direito do mouse sobre a pasta **Operators** e escolha a opção **New Operator**;
4. Na tela que será exibida, no campo **Name**, escreva o nome do operador;
5. No campo **E-mail name**, escreva o seu e-mail;
6. No campo **Net Send Address**, escreva o nome do seu servidor;
7. Aceite todas as outras opções como default e clique em **OK**.

B – Configurando o Database Mail

Para poder configurar o **Database Mail**, é preciso habilitar esta configuração. Para tanto, devemos realizar o seguinte procedimento:

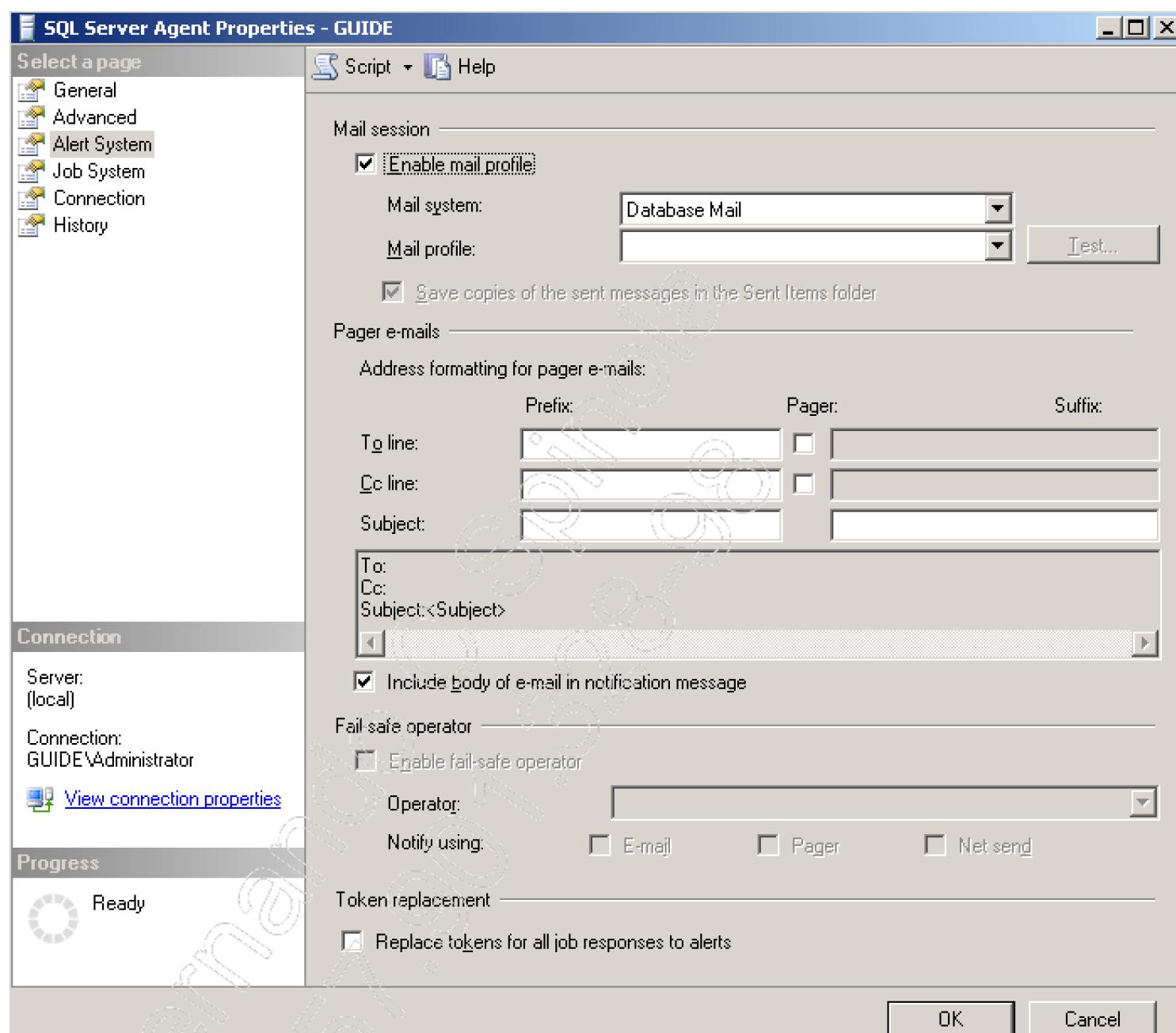
1. Clique no botão **Start**, em seguida, em **All Programs**, depois em **Microsoft SQL Server 2014**, escolha a opção **Microsoft SQL Server Management Studio** e conecte-se no **SQL Server** com a autenticação do Windows;

2. Na tela que será exibida, no **Object Explorer**, clique com o botão direito do mouse sobre o item **SQL Server Agent** e escolha **Properties**, como mostra a próxima imagem:



Automação de tarefas, alertas e operadores

3. Do lado esquerdo da próxima tela, escolha a opção **Alert System** e marque a opção **Enable mail profile**, como ilustra a próxima imagem:



4. Clique no botão **OK**. Será exibida a tela inicial do Microsoft SQL Server Management Studio;
5. Neste ponto, crie um perfil para o uso do **Database Mail**;
6. Expanda a pasta **Management**;
7. Clique com o botão direito do mouse sobre **Database Mail**;
8. Escolha a opção **Configure Database Mail**;
9. Na tela que será exibida, clique em **Next**;
10. Na nova tela, deixe selecionada a opção **Set up Database Mail by performing the following tasks** e clique no botão **Next**;
11. No campo **Profile Name** da nova tela, escreva **DBA**;
12. No campo **Description**, escreva **Primeiro perfil para utilizar o Database Mail**;
13. Do lado direito dessa tela, clique no botão **Add** para acrescentar uma conta de usuário para este perfil;
14. Na tela que será exibida, clique no botão **New Account**;
15. Na próxima tela, no campo **Name**, escreva um nome;
16. No campo **Description**, escreva um cargo ocupado na empresa;
17. No campo **E-mail address**, escreva um e-mail válido que é composto pelo usuário do aluno + @universo.net. Exemplo: **Aluno{nº do servidor}@universo.net**;
18. No campo **Display Name**, escreva como o nome deve aparecer ao enviar um e-mail;

19. No campo **Reply e-mail**, repita o e-mail;
20. No campo **Server name**, escreva o nome ou o IP do servidor SMTP. Você pode obter essa informação com o instrutor em sala de aula. Exemplo: **P{nº da sala}_instrutor@universo.net**;
21. Aceite todas as outras opções como default e clique em **OK**;
22. Clique em **Next**;
23. Clique mais duas vezes em **Next** e, em seguida, em **Finish**;
24. Observe que as configurações foram feitas com sucesso e clique em **Close**;
25. Novamente no **Object Explorer**, clique com o botão direito do mouse sobre **Database Mail** e escolha a opção **Send Test E-Mail**;
26. Na tela que será exibida, no campo **To**, escreva um e-mail para quem desejamos que o teste seja enviado. Neste caso, escreva um e-mail que possa ser aberto para a verificação da mensagem recebida;
27. Clique no botão **Send Test E-Mail** e, em seguida, em **OK**;
28. A seguir, no Microsoft SQL Server Management Studio, na barra de ferramentas, clique em **New Query**;
29. Escreva e execute a procedure a seguir:

```
Exec msdb.dbo.SP_Send_dbmail
@Profile_name = 'DBA',
@Recipients = 'escrever um e-mail que possa ser aberto',
@Body = 'Teste enviado pelo SQL Server 2014',
@Subject = 'Mensagem enviada com sucesso'
```

30. No e-mail escrito, em **@Recipients**, verifique se a mensagem chegou com sucesso.

C – Configurando o primeiro job

Neste exercício, vamos criar um job que executa uma **stored procedure** uma vez por mês e envia uma notificação para o operador (criado anteriormente na parte B do laboratório) todas as vezes que a procedure for executada com sucesso.

Este exercício será executado em quatro partes: preparação do ambiente, configuração do job, execução do job manualmente e visualização do histórico do job.

1ª Parte: Preparação do ambiente

Para preparar o ambiente para a criação e execução deste job, no Microsoft SQL Server Management Studio, clique em **File**, em seguida, em **Open** e abra o **Script_02** da pasta **Capitulo_08**. Esse script fará o seguinte:

- Criará um database chamado **SuperMercado**;
- Dentro deste database serão criadas três tabelas (**Funcionario**, **Pedido** e **Premio**);
- Inserirá algumas linhas de dados nas tabelas **Funcionario** e **Pedido**;
- Criará uma stored procedure chamada **P_GeraPremio**. Esta procedure gera um prêmio para os funcionários nas seguintes condições:
 - Se o funcionário vendeu entre 500.00 e 1000.00 em pedidos, ele receberá um prêmio de 50.00;
 - Se o funcionário vendeu entre 1000.00 e 3000.00 em pedidos, ele receberá um prêmio de 200.00;
 - Se o funcionário vendeu mais de 3000.00 em pedidos, ele receberá um prêmio de 500.00.

1. Com o **Script_02** aberto no Microsoft SQL Server Management Studio, pressione a tecla F5 para executá-lo;
2. Feche o **Script_02**.

2ª Parte: Configuração do job

1. No Object Explorer, expanda SQL Server Agent;
2. Clique com o botão direito do mouse sobre a pasta **Jobs** e escolha a opção **New Job**;
3. No campo **Name**, escreva o nome do job, que, neste caso, é **Job_GeraPremio**;
4. No campo **Description**, escreva **De acordo com os pedidos atendidos, gera um prêmio para os funcionários na última sexta-feira de cada mês**;
5. Do lado esquerdo superior desta tela, clique em **Steps**;
6. Na parte inferior desta tela, clique no botão **New**;
7. No campo **Step name**, escreva **Passo_Único**;
8. No campo **Type**, aceite o default **Transact-SQL script (T-SQL)**;
9. No campo **Database**, selecione **SuperMercado**;
10. No campo **Command**, escreva **Exec P_GeraPremio**;
11. Clique no botão **Parse** para verificar a sintaxe do comando. Deverá surgir uma mensagem indicando sucesso. Clique em **OK**;
12. No final desta tela, clique em **OK** novamente;
13. Do lado esquerdo superior desta tela, clique no botão **Schedules**;
14. Na parte inferior da tela, clique no botão **New**;
15. Na tela que será exibida, no campo **Name**, escreva o nome do **schedule**: **Última_SextaFeira_Mês**;
16. No campo **Schedule Type**, aceite a opção **Recurring**;

SQL 2014 - Módulo III

17. Na área **Frequency** dessa tela, execute:

- No campo **Occurs**, escolha a opção **Monthly**;
- Selecione a opção **The** e escolha **Last** e **Friday**.

18. Na área **Daily frequency**, no campo **Occurs once at**, aceite o default **00:00:00**;

19. Nos outros campos restantes, aceite o default;

20. Clique no botão **OK**;

21. Mais uma vez, do lado esquerdo superior desta tela, escolha a opção **Notifications**;

22. Na tela que será exibida, clique em **E-mail**, escolha o nome do operador que será notificado e, em seguida, escolha **When the job succeeds**;

23. Clique em **Net Send**, escolha o nome do operador que será notificado e, em seguida, escolha **When the job succeeds**;

24. Clique no botão **OK** e aguarde alguns segundos para que o SQL Server execute todas as suas configurações.

3^a Parte: Execução manual do job

1. Antes de executar manualmente este job, verifique que, neste momento, não haja registro algum inserido na tabela **Premio**. Para tanto, no Microsoft SQL Server Management Studio, na barra de ferramentas, clique em **New Query** e escreva o comando a seguir, observando que a tabela **Premio** ainda está vazia:

```
Use SuperMercado  
go  
SELECT * FROM Premio
```

2. Agora, no **Object Explorer**, com o **SQL Server Agent** expandido, expanda a pasta **Jobs**;
3. Clique com o botão direito do mouse sobre o **Job_GeraPremio**;
4. Escolha a opção **Start Job at Step**;
5. Observe que uma nova tela surge mostrando o andamento da execução do job em questão e, logo em seguida, surge a mensagem **Success** na janela, indicando que tudo ocorreu com sucesso;
6. Clique em **Close**;
7. Execute os comandos a seguir novamente, para verificar as tarefas executadas pelo job, e observe que agora existem alguns registros na tabela **Premio**:

```
Use SuperMercado  
go  
SELECT * FROM Premio
```

8. Para completar a verificação de sucesso da execução deste job, abra o e-mail e verifique se a mensagem de sucesso também foi recebida por este meio;
9. Para encerrar, feche a query sem salvar os comandos.

4ª Parte: Visualização do histórico do job

1. Clique com o botão direito do mouse sobre **Job_GeraPremio** e escolha a opção **View History**;
2. Na tela que será exibida, do lado esquerdo superior, clique em **Job History**;
3. Observe que surgem do lado direito da tela, informações sobre as vezes que este job rodou, com as mensagens de cada execução;
4. Clique no botão **Close**.

Laboratório 2

- **Segundo Job**

Neste exercício vamos criar um job com dois schedules. Vamos implementar, no database **SuperMercado**, criado no exercício anterior, a rotina de **backup** apresentada a seguir.

- **Legenda**

F	Backup Full (Backup completo)
D	Backup Diff (Backup diferencial)
L	Backup Log (Backup do Transaction Log)

- **Rotina de backup do database Empresa:**

Horário	Seg	Ter	Qua	Qui	Sex	Sab	Dom
7	F	F	F	F	F	F	F
8	L	L	L	L	L		
9	L	L	L	L	L		
10	L	L	L	L	L		
11	L	L	L	L	L		
12	L	L	L	L	L		
13	D	D	D	D	D	D	
14	L	L	L	L	L		
15	L	L	L	L	L		
16	L	L	L	L	L		
17	L	L	L	L	L		
18	F	F	F	F	F	F	F

Esta rotina de backup nos indica que:

- Às 7h e às 18h, todos os dias, deve-se realizar um backup completo do database **Empresa**;
- Às 13h, de segunda a sábado, deve-se executar um backup diferencial do database **Empresa**;
- De segunda a sexta-feira, das 8h às 12h e das 14h às 17h, deve-se realizar um backup do **Transaction log** de hora em hora.

Além disso, o job que executar esta rotina de backup deverá notificar o operador por e-mail e por **Net Send** quando o job falhar.

Este exercício deverá ser executado em cinco partes: criação de um device de backup, criação do job de backup full, criação do job de backup diferencial, criação do job de backup log, teste dos jobs de backup.

A – Criando um device de backup

1. Utilizando o **Windows Explorer**, crie um diretório (pasta) chamado **C:\BackupSuperMercado**;
2. Abra o Microsoft SQL Server Management Studio e expanda a pasta **Server Objects**;
3. Clique com o botão direito do mouse sobre **Backup Devices** e escolha a opção **New Backup Device**;
4. Na tela que será exibida, no campo **Device Name**, escreva **BackupSuperMercado**;
5. Na área **Destination**, deixe a opção **File** marcada e clique nas reticências (...). Em seguida, escolha **C:\BackupMercado\SuperMercado.bak** e clique em **OK**;
6. No **Object Explorer**, expanda a pasta **Backup Devices** e observe que o device de backup do database **SuperMercado** foi criado.

SQL 2014 - Módulo III

B – Criando job de backup full

1. Expanda o **SQL Server Agent**;
2. Clique com o botão direito sobre **Job** e escolha a opção **New Job**;
3. Na tela que será exibida, no campo **Name**, escreva **Backup Full SuperMercado**;
4. No campo **Description**, escreva **Primeiro Backup Full do database SuperMercado**;
5. Do lado esquerdo superior dessa tela, clique na opção **Steps**;
6. Na parte inferior da tela exibida, clique no botão **New**;
7. No campo **Step name**, escreva **Backup Full**;
8. No campo **Database**, escolha **SuperMercado**;
9. No campo **Command**, escreva:

```
BACKUP DATABASE SuperMercado  
TO BackupSuperMercado  
WITH  
DESCRIPTION = 'Backup Full do Database SuperMercado',  
INIT,  
MEDIANAME = 'Backup SuperMercado',  
NAME = 'Backup Full SuperMercado'
```

10. Clique no botão **Parse**;
11. Deverá aparecer uma mensagem informando que o comando foi verificado com sucesso. Clique no botão **OK**;
12. Clique no botão **OK** novamente;
13. Do lado esquerdo superior desta tela, clique em **Schedules**;

14. Na parte inferior da tela, clique no botão **New**;
15. No campo **Name** da tela que será exibida, escreva **Full_7H**;
16. Na área **Frequency**, no campo **Occurs**, escolha a opção **Daily**;
17. Na área **Daily frequency**, no campo **Occurs once at**, escreva **07:00:00**;
18. Aceite todas as outras opções como default e clique no botão **OK**;
19. Do lado esquerdo superior desta tela, clique na opção **Notifications**;
20. Na tela que será exibida, clique em **E-mail** e escolha um operador para ser notificado **When the job fails**;
21. No campo **Net send**, escolha um operador para ser notificado **When the job fails**;
22. Clique no botão **OK** e, em seguida, em **OK** novamente.

C – Criando job de backup diferencial

1. Clique com o botão direito sobre **Job** e escolha a opção **New Job**;
2. Na tela que será exibida, no campo **Name**, escreva **Backup Diff SuperMercado**;
3. No campo **Description**, escreva **Backup Diferencial do database SuperMercado**;
4. Do lado esquerdo superior desta tela, clique na opção **Steps**;
5. Na parte inferior da tela que será exibida, clique no botão **New**;
6. No campo **Step name**, escreva **Backup Diff**;
7. No campo **Database**, escolha **SuperMercado**;

SQL 2014 - Módulo III

8. No campo **Command**, escreva:

```
BACKUP DATABASE SuperMercado  
TO BackupSuperMercado  
WITH  
DIFFERENTIAL,  
DESCRIPTION = 'Backup Diferencial do database SuperMercado',  
NOINIT,  
MEDIANAME = 'Backup SuperMercado',  
NAME = 'Backup Diferencial SuperMercado'
```

9. Clique no botão **Parse**;

10. Deverá aparecer uma mensagem indicando que o comando foi verificado com sucesso. Clique no botão **OK**;

11. Clique no botão **OK** novamente;

12. Do lado esquerdo superior desta tela, clique em **Schedules**;

13. Na parte inferior da tela, clique no botão **New**;

14. No campo **Name** da tela que será exibida, escreva **Diff_13H**;

15. Na área **Frequency**, no campo **Occurs**, escolha a opção **Daily**;

16. Na área **Daily frequency**, no campo **Occurs once at**, escreva **13:00:00**;

17. Aceite todas as outras opções como default e clique no botão **OK**;

18. Do lado esquerdo superior desta tela, clique na opção **Notifications**;

19. Na tela que será exibida, clique em **E-mail** e escolha um operador para ser notificado **When the job fails**;

20. No campo **Net send**, escolha um operador para ser notificado **When the job fails**;

21. Clique no botão **OK**.

D – Criando job de backup log

1. Clique com o botão direito sobre **Job** e escolha a opção **New Job**;
2. Na tela que será exibida, no campo **Name**, escreva **Backup Log SuperMercado**;
3. No campo **Description**, escreva **Backup Log do database SuperMercado**;
4. Do lado esquerdo superior desta tela, clique na opção **Steps**;
5. Na parte inferior da tela que será exibida, clique no botão **New**;
6. No campo **Step name**, escreva **Backup Log**;
7. No campo **Database**, escolha **SuperMercado**;
8. No campo **Command**, escreva:

```
BACKUP LOG SuperMercado  
TO BackupSuperMercado  
WITH  
DESCRIPTION = 'Backup Log do Database SuperMercado',  
NOINIT,  
MEDIANAME = 'Backup SuperMercado',  
NAME = 'Backup Log SuperMercado'
```

9. Clique no botão **Parse**;
10. Deverá aparecer uma mensagem informando que o comando foi verificado com sucesso. Clique no botão **OK**;
11. Clique no botão **OK** novamente;
12. Do lado esquerdo superior dessa tela, clique em **Schedules**;
13. Na parte inferior da tela, clique no botão **New**;
14. No campo **Name** da tela que será exibida, escreva **Log_8_12H**;
15. Na área **Frequency**, no campo **Occurs**, escolha a opção **Daily**;

SQL 2014 - Módulo III

16. Na área **Daily frequency**, no campo **Occurs every**, escolha **1** no primeiro campo, e **hour(s)** no segundo campo. Isso significa que esta tarefa será executada a cada uma hora;
17. No campo **Starting at**, escreva **8:00:00**;
18. No campo **Ending at**, escreva **12:00:00**;
19. Clique no botão **OK**;
20. Clique no botão **New** novamente;
21. No campo **Name** da tela que será exibida, escreva **Log_14_17H**;
22. Na área **Frequency**, no campo **Occurs**, escolha a opção **Daily**;
23. Na área **Daily frequency**, no campo **Occurs every**, escolha **1** no primeiro campo, e **hour(s)** no segundo campo. Isso significa que esta tarefa será executada a cada uma hora;
24. No campo **Starting at**, escreva **14:00:00**;
25. No campo **Ending at**, escreva **17:00:00**;
26. Clique no botão **OK**;
27. Do lado esquerdo superior desta tela, clique na opção **Notifications**;
28. Na tela que será exibida, clique em **E-mail** e escolha um operador para ser notificado **When the job fails**;
29. No campo **Net send**, escolha um operador para ser notificado **When the job fails**;
30. Clique no botão **OK**.

E – Testando os jobs de backup

1. Expanda **SQL Server Agent e Jobs**, clique com o botão direito do mouse sobre **Backup Full SuperMercado** e escolha a opção **Start Job at Step**;
2. Note que surge outra tela mostrando as etapas de execução do job. O job deve ter rodado com sucesso. Clique no botão **Close**;
3. Clique com o botão direito do mouse sobre a **Backup Diff SuperMercado** e escolha a opção **Start Job at Step**;
4. Note que surge outra tela mostrando as etapas de execução do job. O job deve ter rodado com sucesso. Clique no botão **Close**;
5. Clique com o botão direito do mouse sobre **Backup Log SuperMercado** e escolha a opção **Start Job at Step**;
6. Note que surge outra tela mostrando as etapas de execução do job. O job deve ter rodado com sucesso. Clique no botão **Close**.

Laboratório 3

- **Terceiro job**

Neste exercício, vamos configurar um job que executa duas tarefas (dois **steps**):

- Transferir os dados de um arquivo texto para uma tabela;
- Fazer o backup do database.

Este exercício deverá ser realizado em cinco partes: preparação do ambiente para a configuração do job, configuração do job, execução do job, verificação da transferência de dados, verificação da execução do backup.

A – Preparando o ambiente para a configuração do job

Para configurar a tarefa de transferência de dados, precisamos antes criar um database e, dentro dele, criar a tabela que receberá os dados transferidos. Precisamos também criar o device de backup que vai armazenar os backups feitos neste database. Para preparar este ambiente, siga este passo a passo:

1. Clique no botão **Start**, em seguida em **All Programs**, depois em **Microsoft SQL Server 2014** e em **SQL Server Management Studio**;
2. Conecte-se com a autenticação do Windows;
3. Na barra de ferramentas, clique no botão **New Query**;
4. No menu de opções, clique em **File**, em seguida, em **Open** e em **File** novamente. Abra o **Script_04** localizado na pasta **Capítulo_08**. Este script cria o database **Produção** e, dentro dele, a tabela **Produto**. Ele cria também o device de backup.

B – Configurando o job

Vamos agora configurar um job que executa duas tarefas (dois steps). Para tanto, siga os passos descritos adiante:

1. Ainda no SQL Server Management Studio, do lado esquerdo da tela, no **Object Explorer**, expanda **SQL Server Agent**;
2. Clique com o botão direito do mouse sobre **Jobs** e escolha a opção **New Job**;
3. No campo **Name** da tela que será exibida, escreva **JobDoisPassos**;
4. No campo **Description**, escreva **Este Job executa uma transferência de dados e um backup**;
5. Do lado esquerdo superior desta tela, clique em **Steps**;
6. Na parte inferior da tela que será exibida, clique no botão **New**;
7. Na tela que será exibida, no campo **Step Name**, escreva **Passo_01**;
8. No campo **Type**, escolha a opção **Operating system (CmdExec)**;
9. Copie o arquivo **GeraProdutos.cmd** e **Produtos.txt** da pasta **C:\SQLServer2014 - Módulo III\Capítulo_08** para a raiz do **C:**;
10. No campo **Command**, escreva: **C:\GeraProdutos.cmd** e clique no botão **OK**;
11. Clique novamente no botão **New**;
12. No campo **Step Name**, escreva **Passo_02**;
13. No campo **Database**, escolha **Producao**;
14. No campo **Command**, escreva:

BACKUP DATABASE Producao TO BackupProducao

15. Clique no botão **Parse**. Deve aparecer uma mensagem informando que o comando foi escrito corretamente. Em seguida, clique no botão **OK**;
16. Clique no botão **OK** novamente;
17. Do lado esquerdo superior da tela, clique na opção **Schedules** e, no final da tela, escolha a opção **New**;
18. No campo **Name** da tela que será exibida, escreva **SabadoMeiaNoite**;
19. Na área **Frequency**, mantenha **Weekly** no campo **Occurs**, marque a opção **Saturday** e clique no botão **OK**;
20. Do lado esquerdo superior desta tela, escolha a opção **Notifications**;
21. Na tela que será exibida, clique na opção **E-mail** e escolha o nome de um dos operadores. Deixe marcada a opção **When the Job fails**;
22. Clique na opção **Net Send** e escolha o nome de um operador. Deixe marcada a opção **When the job fails**;
23. Clique no botão **OK**.

C - Executando o job

Para executar este job manualmente, faça o seguinte:

1. Ainda utilizando o SQL Server Management Studio, expanda **SQL Server Agent e Jobs**;
2. Clique com o botão direito do mouse sobre o job cujo nome é **JobDoisPassos** e escolha a opção **Start Job at Step**;
3. Clique na opção **Start** na tela que será exibida;
4. O job deve executar com sucesso. Em seguida, clique em **Close**.

D – Verificando a transferência de dados

1. Na barra de ferramentas, clique na opção **New Query**;
2. Escreva e execute os comandos a seguir:

```
Use Producao  
go  
SELECT * FROM Produto
```

3. O SQL Server deve mostrar a tabela **Produto** com 21 registros inseridos.

E – Verificando a execução do backup

1. Do lado esquerdo da tela, no **Object Explorer**, expanda a pasta **Server Objects**;
2. Expand a pasta **Backup Devices**;
3. Clique com o botão direito do mouse sobre o device **BackupProducao** e escolha a opção **Properties**;
4. Do lado esquerdo superior desta tela, clique na opção **Media Contents** e observe que, neste device de backup, há um backup full do database **Producao**;
5. Clique no botão **OK**.

Laboratório 4

A – Criando uma categoria para ser atribuída aos jobs

1. Utilizando o **Microsoft SQL Server Management Studio**, do lado esquerdo da tela, no **Object Explorer**, expanda **SQL Server Agent**;
2. Clique com o botão direito sobre a pasta **Jobs** e escolha a opção **Manage Job Categories**;
3. Na tela que será exibida, clique no botão **Add**;
4. No campo **Name** da nova tela que será exibida, escreva **Impacta**;
5. Clique na opção **Show all jobs**. Neste momento, aparecem na tela todos os jobs que estão sem uma categoria definida;
6. Clique no campo **Select** de todos os jobs;
7. Clique em **OK**;
8. Na tela seguinte, clique no botão **Refresh**;
9. Mova a barra de rolagem vertical para baixo e observe que agora a categoria **Impacta de Jobs** faz parte da lista de categorias. Note que, do lado esquerdo, no campo **Number of jobs in category**, esta tela mostra a quantidade de jobs existentes desta categoria;
10. Clique no botão **Cancel**;
11. Do lado esquerdo da tela, no **Object Explorer**, expanda a pasta **Jobs**;
12. Clique com o botão direito do mouse sobre um dos jobs da lista e escolha a opção **Properties**;
13. Na tela que será exibida, no campo **Category**, observe que está escrito **Impacta**, assim como em todos os jobs configurados até o presente exercício;
14. Clique no botão **OK**;
15. Feche a pasta **Jobs**;
16. Feche o **SQL Server Agent**.

Laboratório 5

Neste exercício, vamos configurar um alerta sobre o erro de número 9002. Este erro acontece quando um arquivo de log enche. Sendo assim, vamos criar um database que não permite o crescimento automático do arquivo de log e, dentro dele, vamos criar uma **table** e uma **stored procedure** que executa uma inclusão de dados nesta tabela até que o log encha. Quando o log encher, o SQL Server emitirá o erro de número 9002 e o alerta será acionado. Este alerta executará um job que realizará um backup do log, para que o database possa continuar a ser utilizado pelos usuários.

Este exercício será executado em seis etapas: preparação do ambiente, criação do job que executa o backup do log, criação do Alert, geração do Alert, observação do Application Log do Event Viewer e do e-mail enviado pelo alerta.

A – Preparando o ambiente para a execução do exercício

Vamos agora executar o **Script_05** da pasta **Capítulo_08** para preparar o ambiente para a execução deste exercício. Este script executa as seguintes tarefas:

- Cria o database **Clients**;
 - No database **Clients**, cria a tabela **Cliente**;
 - Cria uma procedure chamada **P_InsereCliente**, que insere registros na tabela **Cliente**.
1. No Microsoft SQL Server Management Studio, no menu de opção, clique em **File**, em seguida, em **Open** e em **File** novamente;
 2. Abra o **Script_05** e pressione a tecla F5 para executá-lo.

B – Criando o job que executa o backup do log

Este job que será criado agora será acionado por um Alert quando o arquivo de log estiver cheio.

1. Expanda o **SQL Server Agent**;
2. Clique com o botão direito do mouse sobre **Jobs** e escolha a opção **New Job**;
3. Na tela que será exibida, no campo **Name**, escreva **Job_BackupCliente**;
4. No campo **Category**, escolha **Impacta**;
5. No campo **Description**, escreva **Realiza um Backup do Transaction Log do Database Clientes**;
6. Do lado esquerdo superior desta tela, escolha a opção **Steps**;
7. Clique no botão **New**;
8. No campo **Step Name** da tela que será exibida, escreva **BackupCliente**;
9. No campo **Database**, escolha **Clientes**;
10. No campo **Command**, escreva:

```
BACKUP DATABASE Clientes  
TO DISK = 'C:\BackupClientes\BackupCliente.bak'
```

11. Clique no botão **Parse** para verificar a sintaxe do comando;
12. Deve aparecer uma mensagem indicando sucesso. Clique no botão **OK**;
13. Clique em **OK** novamente.

C – Criando o Alert

1. No lado superior esquerdo da tela, clique em **Alerts**;
2. Clique no botão **Add**;
3. No campo **Name** da tela que será exibida, escreva **LogCheio**;
4. No campo **Database Name**, escolha **Clientes**;
5. Selecione o campo **Error Number** e escreva **9002**;
6. Do lado esquerdo superior desta tela, clique em **Response**;
7. Na tela que será exibida, clique em **Notify Operators**;
8. Clique nas opções **E-mail** e **Net Send** dos operadores que desejar que sejam notificados quando este alerta ocorrer;
9. Clique em **OK**;
10. Clique em **OK** novamente.

D – Gerando o Alert

1. Ainda no Microsoft SQL Server Management Studio, execute a procedure **P_InsereCliente** para ver o erro **9002** acontecendo e o alerta **LogCheio** sendo acionado. Para tanto, escreva e execute o comando a seguir:

```
Exec P_InsereCliente
```

2. Observe que o **SQL Server** insere alguns registros na tabela **Cliente** e, em seguida, aparece a seguinte mensagem de erro:

```
Msg 9002, Level 17, State 2, Procedure P_InsereCliente, Line 12  
The transaction log for database 'Clientes' is full due to 'LOG_BACKUP'.
```

3. Logo após aparecer a mensagem de erro anterior, note que surge na tela a notificação gerada pelo alerta existente sobre o erro de número **9002**;
4. Para corrigir o erro de LOG cheio execute o **Script_06**.

E – Observando o Application Log do Event Viewer

Um alerta só será acionado se a notificação da ocorrência do respectivo erro for registrada no **Log de Aplicações** do Event Viewer no Windows. Para fazer esta verificação, siga este passo a passo:

1. No menu do Windows, clique no botão **Start**, abra **All Programs**, **Administrative Tools** e escolha **Event Viewer**;
2. Na tela que será exibida, do lado esquerdo superior, clique em **Applications and Services Logs**. Note que há um erro do **SQL Server** registrado neste log de aplicações;
3. Feche o **Event Viewer**.

F – Observando o e-mail enviado pelo alerta

1. Para verificar que o SQL Server notificou o operador também por correio eletrônico, abra o e-mail do referido operador e veja a mensagem enviada pelo SQL Server;
2. Feche o **Script_05** e o **Script_06**.

Laboratório 6

Neste exercício, vamos configurar um alerta que será acionado em uma das condições do **Performance Monitor**.

Queremos monitorar a quantidade de transações que serão executadas concurrentemente. Neste exercício, se duas ou mais transações forem iniciadas concurrentemente, vamos fazer o **SQL Server** enviar uma mensagem para o operador avisando-o do que aconteceu.

O exercício será executado em três etapas: configuração do alerta, geração das transações concurrentemente e verificação da ocorrência do alerta.

Para completar esta tarefa, siga este passo a passo:

A – Configurando o alerta

1. No Microsoft SQL Server Management Studio, expanda **SQL Server Agent**;
2. Clique com o botão direito do mouse sobre a pasta **Alerts** e escolha **New Alert**;
3. Na tela que será exibida, execute as seguintes tarefas:
 - No campo **Name**, escreva **Alerta_Transações**;
 - No campo **Type**, selecione **SQL Server performance condition alert**;
 - No campo **Object**, escolha **Databases**;
 - No campo **Counter**, escolha **Active Transactions**;
 - No campo **Instance**, escolha **Clientes**;
 - No campo **Alert if counter**, escolha **rises above**;
 - No campo **Value**, escreva **1**.

4. Do lado esquerdo superior dessa tela, clique na opção **Response**;
5. Cheque a opção **Notify Operators**;
6. Novamente do lado esquerdo superior desta tela, clique em **Options**;
7. Na tela que será exibida, na área **Include alert error text in**, selecione **E-mail** e **Net Send**;
8. No campo **Additional notification message to send**, escreva **Seu sistema está executando duas ou mais transações concorrentemente**;
9. No campo **Delay Between Responses**, escreva **0 em minutes e 0 em seconds**;
10. Clique no botão **OK**.

B – Gerando as transações concorrentemente

1. Na barra de ferramentas, clique na opção **New Query**;
2. Na barra de menus, clique na opção **File** e, em seguida, clique em **Open** e em **File** novamente;
3. Abra o arquivo **Script_07** da pasta **Capitulo_08**, conectando-se ao **SQL Server** com a autenticação do Windows. Em seguida, pressione F5 para executá-lo;
4. Escreva e execute o seguinte comando:

```
Use Clientes
Go

BEGIN TRANSACTION
    UPDATE Funcionario
    SET Nome_Func = 'Ana carolina'
    WHERE Cod_Func = 1
```

5. Na barra de ferramentas, clique em **New Query** para que seja aberta uma nova sessão;

6. Na nova conexão, escreva e execute o comando a seguir:

```
Use Clientes
go
BEGIN TRANSACTION
    UPDATE Funcionario
    SET Nome_Func = 'José Pedro'
    WHERE Cod_Func = 2
```

7. Aguarde alguns segundos para que surja na tela a notificação enviada pelo alerta;

8. Observe o conteúdo da mensagem anterior e clique no botão **OK** para fechá-la;

9. Para encerrar as transações, nas duas sessões, escreva e execute o comando **ROLLBACK TRANSACTION**;



Observe que, enquanto pelo menos uma das transações não for encerrada, o alerta continuará a ser emitido.

10. Feche todas as sessões de comando que foram abertas.

C – Verificando a ocorrência do alerta

1. Para verificar quantas vezes aquele alerta ocorreu, no **Object Explorer**, expanda **SQL Server Agent**;
2. Expanda **Alerts**;
3. Clique com o botão direito do mouse sobre o **Alerta_Transações** e escolha a opção **Properties**;
4. Do lado esquerdo superior da tela que será exibida, clique em **History**;
5. No campo **Number of occurrences** da tela exibida, observe o número de vezes que o alerta foi acionado;
6. Clique no botão **OK**.

Laboratório 7

Neste exercício, vamos criar um alerta sobre um número de erro definido pelo usuário.

Para realizar este exercício, na preparação do ambiente, vamos executar as seguintes tarefas:

- Criação de uma mensagem de erro definida pelo usuário de número **50001**;
- Criação de um database chamado **Vendas**;
- Criação dos seguintes objetos no database **Vendas**:
 - Três tabelas chamadas **Funcionario**, **Pedido** e **Premio**;
 - Uma **Procedure** chamada **P_EnviaMensagem**;
 - Um **trigger** de inclusão na tabela **Pedido** chamado **T_GeraPremio**.
- Criação do erro **50001**;
- Criação de um alerta sobre o erro **50001**.

Depois de configurado, este exercício funcionará da seguinte maneira: todas as vezes que um pedido com valor igual ou superior a 500.00 for inserido na tabela **Pedido**, o **trigger** de inclusão dará um prêmio de 50.00 para o funcionário que atendeu esse pedido, executará uma **procedure** que enviará por e-mail (com a **extended stored procedure xp_send_dbmail**) a posição geral da premiação para o administrador e gerará o erro **50001**, registrando-o no log de aplicações do **Event Viewer**. Assim, o **Alerta_50001** será acionado e enviará, por e-mail e por **Net Send**, uma mensagem para o **Operador**, avisando-o de que mais um funcionário recebeu a premiação.

Este laboratório será executado em quatro partes: preparação do ambiente, criação do alerta sobre o erro 50001, teste das configurações e verificação do Application Log do Event Viewer.

A – Preparando o ambiente

Vamos executar o **Script_10** da pasta **Capitulo_08**, que criará:

- O Database **Vendas**;
- A mensagem de erro de número **50001**;
- As tabelas **Funcionario**, **Pedido** e **Premio**;
- A procedure **P_EnviaMensagem**;
- O trigger **T_GeraPremio**.

1. No Microsoft SQL Server Management Studio, na barra de ferramentas, clique em **New Query**. Conecte-se com a autenticação do Windows;
2. No menu de opções, clique em **File**, em seguida, em **Open** e depois em **File** novamente. Abra o **Script_10** da pasta **Capitulo_08**;
3. Pressione F5 para executar esse script.

B – Criando o alerta sobre o erro 50001

1. Do lado esquerdo da tela, no **Object Explorer**, expanda **SQL Server Agent**;
2. Clique com o botão direito do mouse sobre a pasta **Alerts** e escolha a opção **New Alerts**;
3. Na tela exibida, execute:
 - No campo **Name**, escreva **Alerta_50001**;
 - No campo **Database name**, selecione **Vendas**;
 - Selecione a opção **Error Number** da sessão **Event alert definition** e escreva **50001**.

4. No lado esquerdo superior desta tela, clique na opção **Response**;
5. Na tela que será exibida, cheque a opção **Notify Operators**;
6. No campo **Operator** da seção **Operator list**, selecione as opções **E-mail** e **Net Send** para um dos operadores da sua lista;
7. No lado esquerdo superior desta tela, escolha a opção **Options**;
8. Na seção **Include alert error text in**, cheque as opções **E-mail** e **Net send**;
9. No campo **Additional notification message to send**, escreva: **Mais uma premiação foi concedida a um dos funcionários por ter atendido a um pedido de valor maior ou igual a 500.00**;
10. Clique no botão **OK**.

C – Testando as configurações

1. Na barra de ferramentas, clique na opção **New Query**;
2. No menu de opções, escolha a opção **File**, em seguida, **Open** e depois **File** novamente;
3. Abra o **Script_11** da pasta **Capitulo_08** para fazer alguns testes;
4. Com o **Script_11** aberto, selecione apenas os comandos a seguir e pressione F5 para executá-los:

```
-- Acessando o database Vendas
Use Vendas
go
-- Lendo o conteúdo das tabelas
SELECT * FROM Funcionario
SELECT * FROM Pedido
SELECT * FROM Premio
```

5. Observe a resposta: A tabela **Funcionario** possui dados; As tabelas **Pedido** e **Premio** estão vazias;

6. Insira um pedido cujo valor é 1500.00 e observe o que acontece. Para tanto, execute apenas o comando a seguir:

```
INSERT Pedido VALUES(2, getdate(), 1500.00)
```

7. Aguarde alguns segundos até que o SQL Server execute o comando e observe a resposta;

8. Observe que o erro **50001** acionou o alerta;

9. Para fechar a mensagem, clique no botão **OK**;

 Insira os outros pedidos (executando o **Script_12**) para verificar que, quando o valor é inferior a 500.00, o alerta não é acionado e o funcionário em questão não recebe o prêmio. Inclua também os pedidos cujo valor é superior a 500.00, para ver todo o mecanismo configurado ocorrendo novamente.

10. Abra o e-mail do operador selecionado para receber a mensagem do alerta e verifique a chegada da mensagem;

11. No **Microsoft SQL Server Management Studio**, execute novamente os comandos a seguir:

```
SELECT * FROM Funcionario  
SELECT * FROM Pedido  
SELECT * FROM Premio
```

12. Observe que um pedido de 1500.00 foi inserido para o funcionário de código 2 e que ele recebeu um prêmio de 50.00.

D – Verificando o application log do Event Viewer

1. Abra o **Application Log** do **Event Viewer**. Clique no botão **Start**, escolha a opção **All Programs**, em seguida selecione a opção **Administrative Tools** e escolha a opção **Event Viewer**;
2. Na tela que será exibida, clique em **Application and Services Logs**;
3. Do lado direito da tela, no painel de detalhes, aplique um duplo-clique na primeira mensagem de erro que aparece;
4. Observe que a mensagem refere-se ao erro **50001**;
5. Clique no botão **OK** para fechar esta tela;
6. Feche o **Event Viewer**;
7. No Microsoft SQL Server Management Studio, feche a conexão com o **Script_10**, **Script_11** e **Script_12** se eles ainda estiverem abertos.

Replicação e distribuição de dados

9

- ✓ Transação distribuída;
- ✓ Replicação;
- ✓ Escolhendo a estratégia para deposição de dados;
- ✓ Replicação de dados no SQL Server;
- ✓ Tipos de assinaturas;
- ✓ Agentes de replicação;
- ✓ Tipos de publicação;
- ✓ Cenário de replicação;
- ✓ Restrições de replicação.

9.1. Introdução

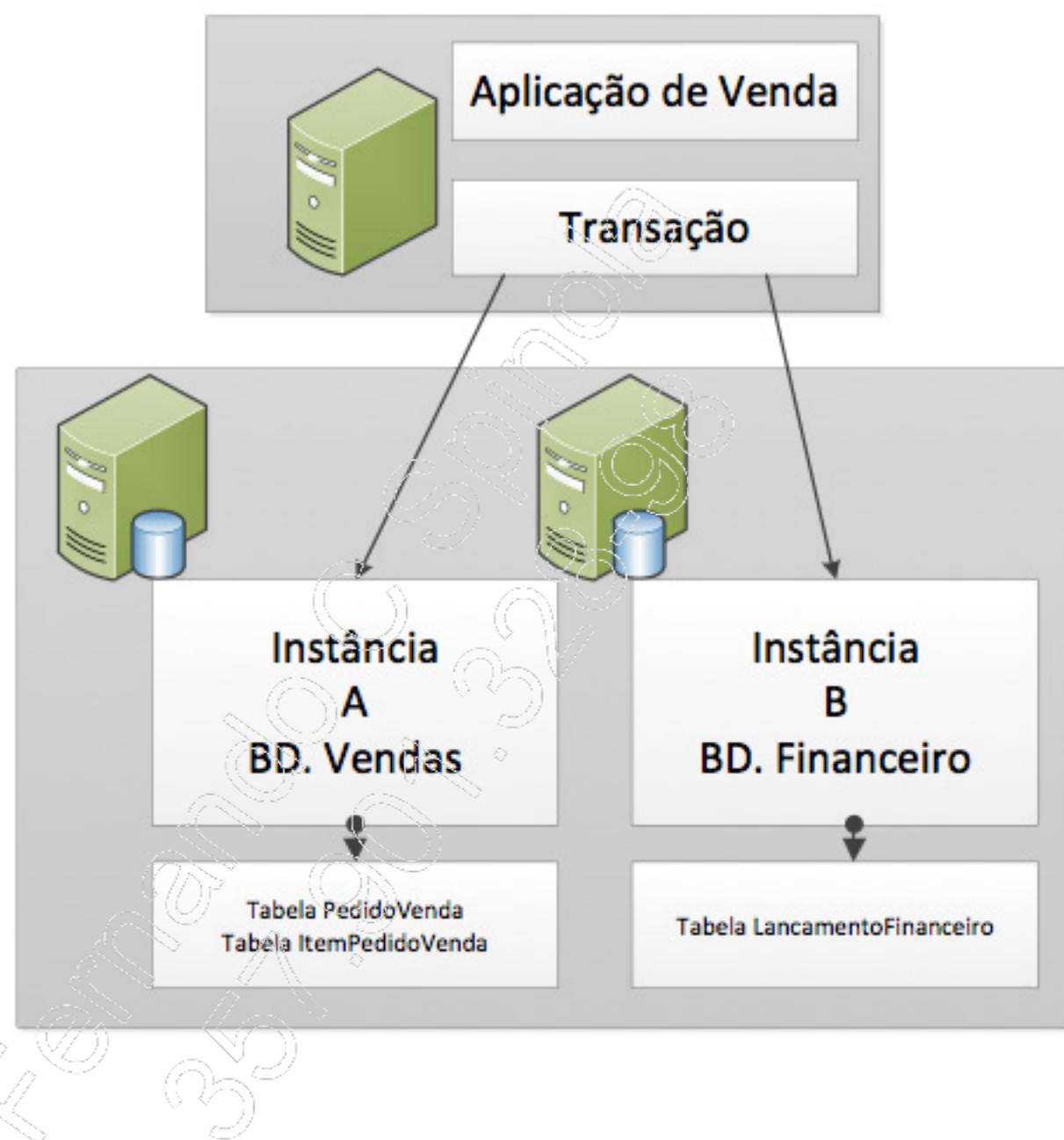
Replicação e distribuição são duas estratégias que podemos dispor para realizar transações entre bancos de dados e/ou instâncias distintas. A seguir, vamos conhecer esses mecanismos de deposição de dados, como são cada um deles e como devemos utilizá-los da maneira mais apropriada.

9.2. Transação distribuída

Transação distribuída é a operação realizada em uma transação que demanda dois ou mais servidores de banco de dados. Ela ocorre individualmente em cada banco, mas está ligada a uma transação única. Ao salvarmos a transação, os bancos de dados participantes deverão efetivá-la e reportar essa tarefa. Esse mecanismo é conhecido como **Two-phase commit**. O foco das transações distribuídas não é a realização da cópia de dados, mas a distribuição dos dados entre os bancos participantes.

Vamos imaginar o seguinte exemplo para entender esse conceito: uma empresa possui dois bancos de dados em instâncias diferentes. O primeiro banco de dados é o de **vendas** e o segundo é o **financeiro**. Todas as vendas são registradas no banco de dados de vendas e todas as informações resultantes dessa ação são armazenadas no banco de dados financeiro. Como se trata de instâncias diferentes (residentes em servidores distintos ou não), a transação é iniciada pelo sistema de vendas alimentando o banco de dados de vendas e, depois que a venda é realizada, os dados são armazenados no banco de dados financeiro.

Vejamos esta transação:



9.3. Replicação

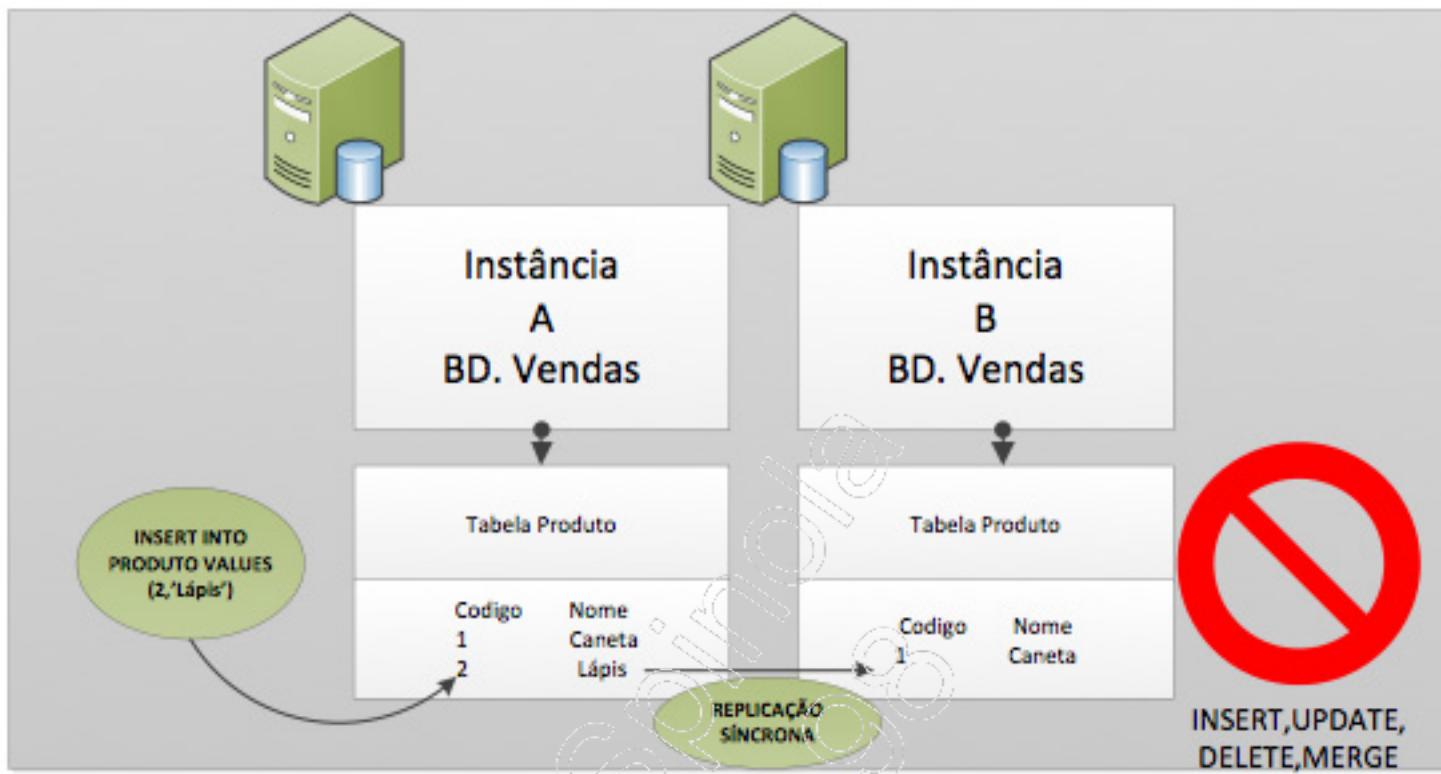
A replicação de dados é outra técnica utilizada para copiar dados no mesmo banco de dados ou em outros bancos de dados e em outras instâncias. Essa estratégia é diferente da distribuição, já que o foco da replicação é a cópia dos dados. Em caso de indisponibilidade da origem das informações, essa técnica estará disponível. Existem vários tipos de replicação de dados, entre eles podemos destacar os seguintes:

- Síncrona unidirecional;
- Síncrona bidirecional;
- Assíncrona unidirecional;
- Assíncrona bidirecional.

9.3.1. Síncrona unidirecional

A replicação síncrona unidirecional pode ser feita através da cópia dos dados modificados de um banco de dados (principal) para outro (réplica). No banco de dados replicado, os dados não sofrem nenhum tipo de manutenção (INSERT, UPDATE, DELETE ou MERGE), apenas consultas. Este tipo de replicação é útil para atualização de dados que não devem ser modificados. Por exemplo, uma empresa que possui uma rede de lojas. Cada uma das lojas terá seu cadastro de produtos atualizados a partir da matriz. As lojas somente poderão vender os produtos que forem informados pela matriz. No momento em que a matriz cadastrar um produto, este será imediatamente disponibilizado para as filiais com um mínimo de tempo possível (tempo necessário para a replicação).

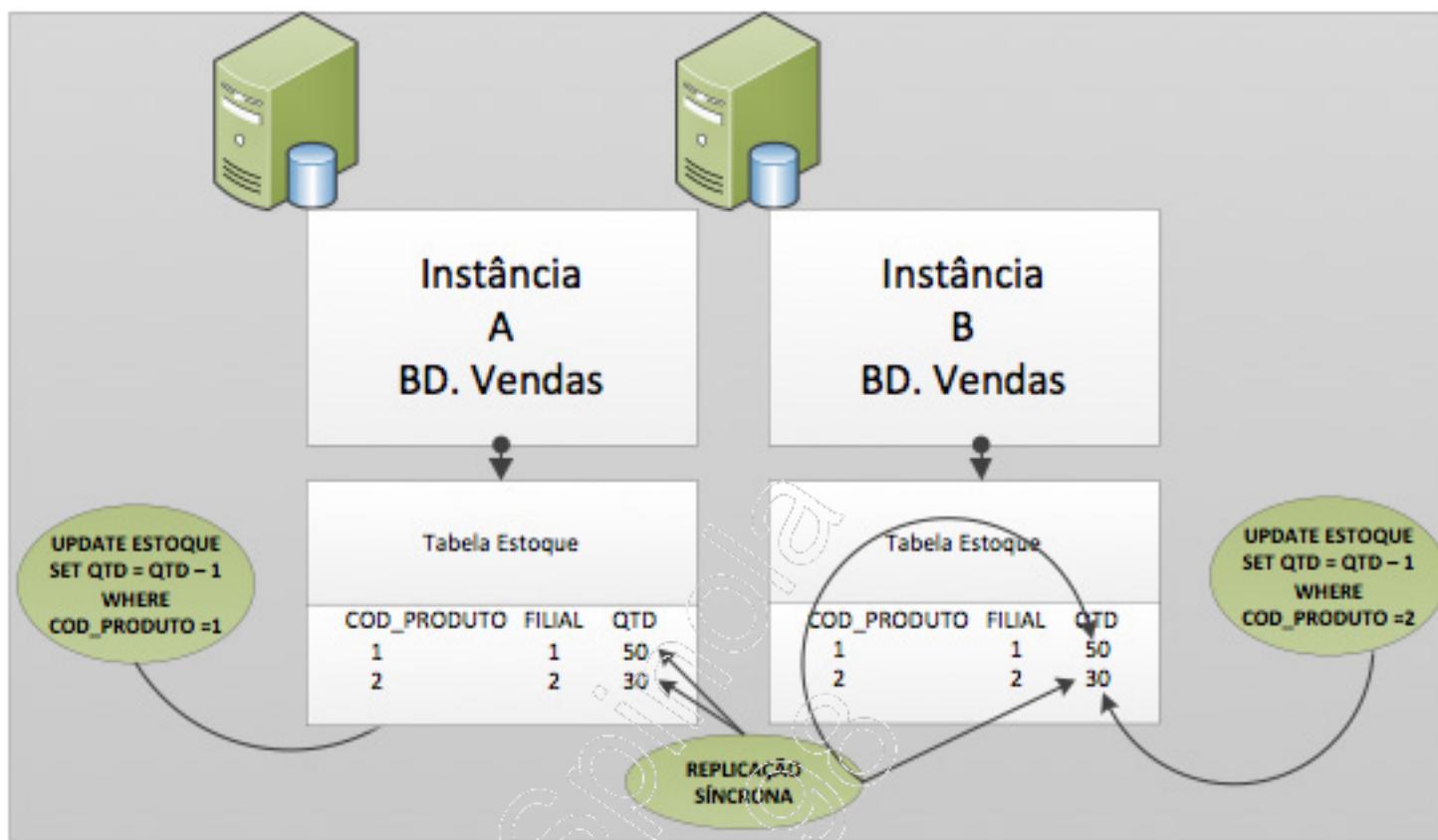
Vejamos o esquema de uma replicação síncrona unidirecional:



9.3.2. Síncrona bidirecional

A replicação síncrona bidirecional pode ser feita através da cópia dos dados modificados em um banco de dados (principal) para outro (réplica). Neste segundo banco, os dados podem sofrer manutenção (INSERT, UPDATE, DELETE ou MERGE) e consultas. Este tipo de replicação é útil para manter os dados atualizados e disponíveis em todos os bancos de dados. Por exemplo, uma empresa que possui filiais em todo o país e seus estoques são descentralizados, mas cada filial precisa disponibilizar o seu estoque para as demais filiais, o que implica em atualização on-line das informações em todas as filiais.

Vejamos o esquema de uma replicação síncrona bidirecional:

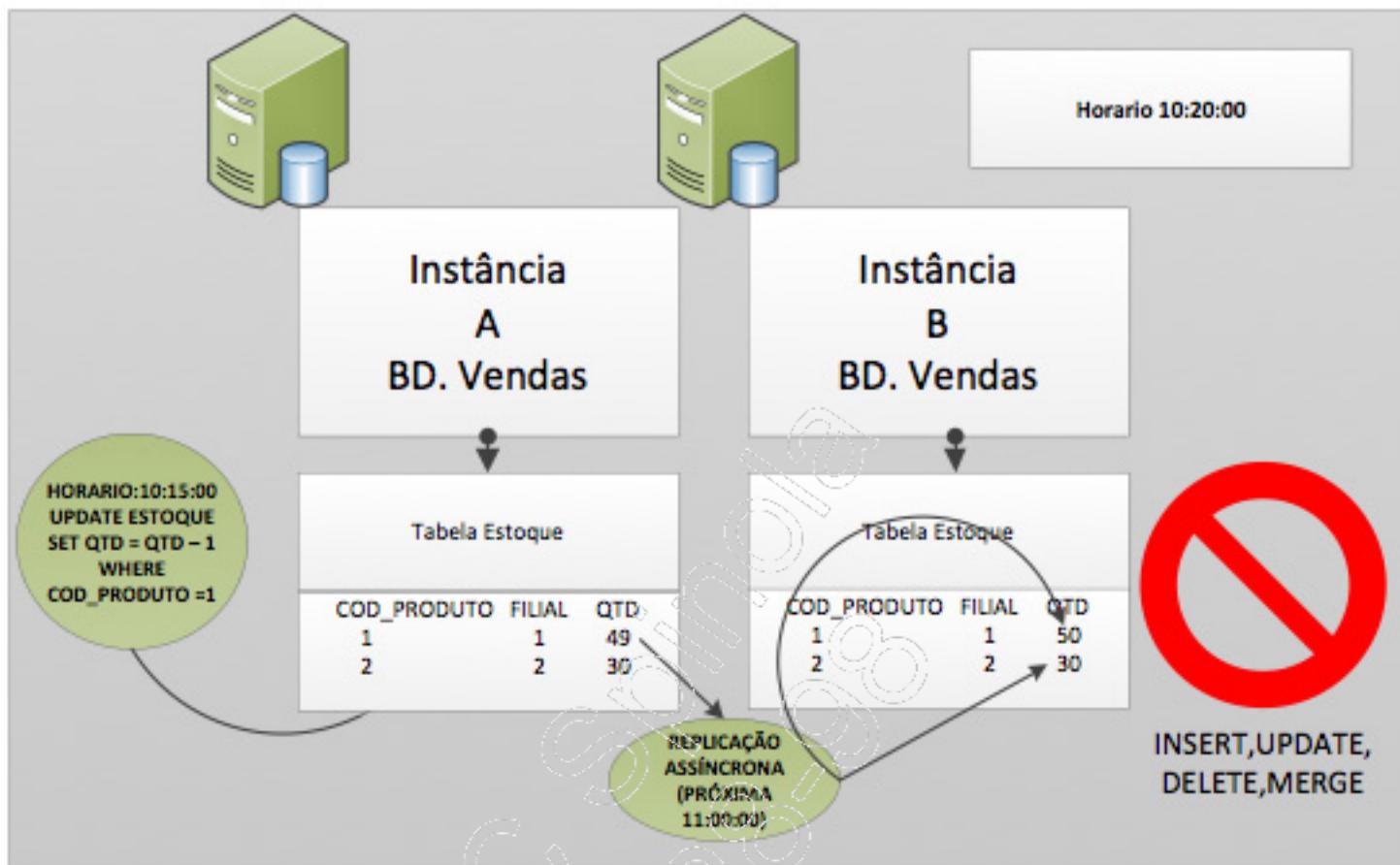


9.3.3. Assíncrona unidirecional

A replicação assíncrona unidirecional pode ser feita através da cópia dos dados modificados em um banco de dados (principal) para outro (réplica). Neste último, os dados não podem sofrer manutenção (INSERT, UPDATE, DELETE ou MERGE), apenas consultas.

Trata-se do tipo de comunicação mais simples, pois a informação replicada não deve ser atualizada, exceto pelo mecanismo de replicação. Logo, as tabelas replicadas são apenas utilizadas em consultas e isso diminui os possíveis conflitos de replicação que são possíveis na replicação assíncrona bidirecional. Um exemplo deste tipo de replicação poderia ser a atualização do preço de produtos em um site de venda.

A seguir, temos um esquema de uma replicação assíncrona unidirecional:



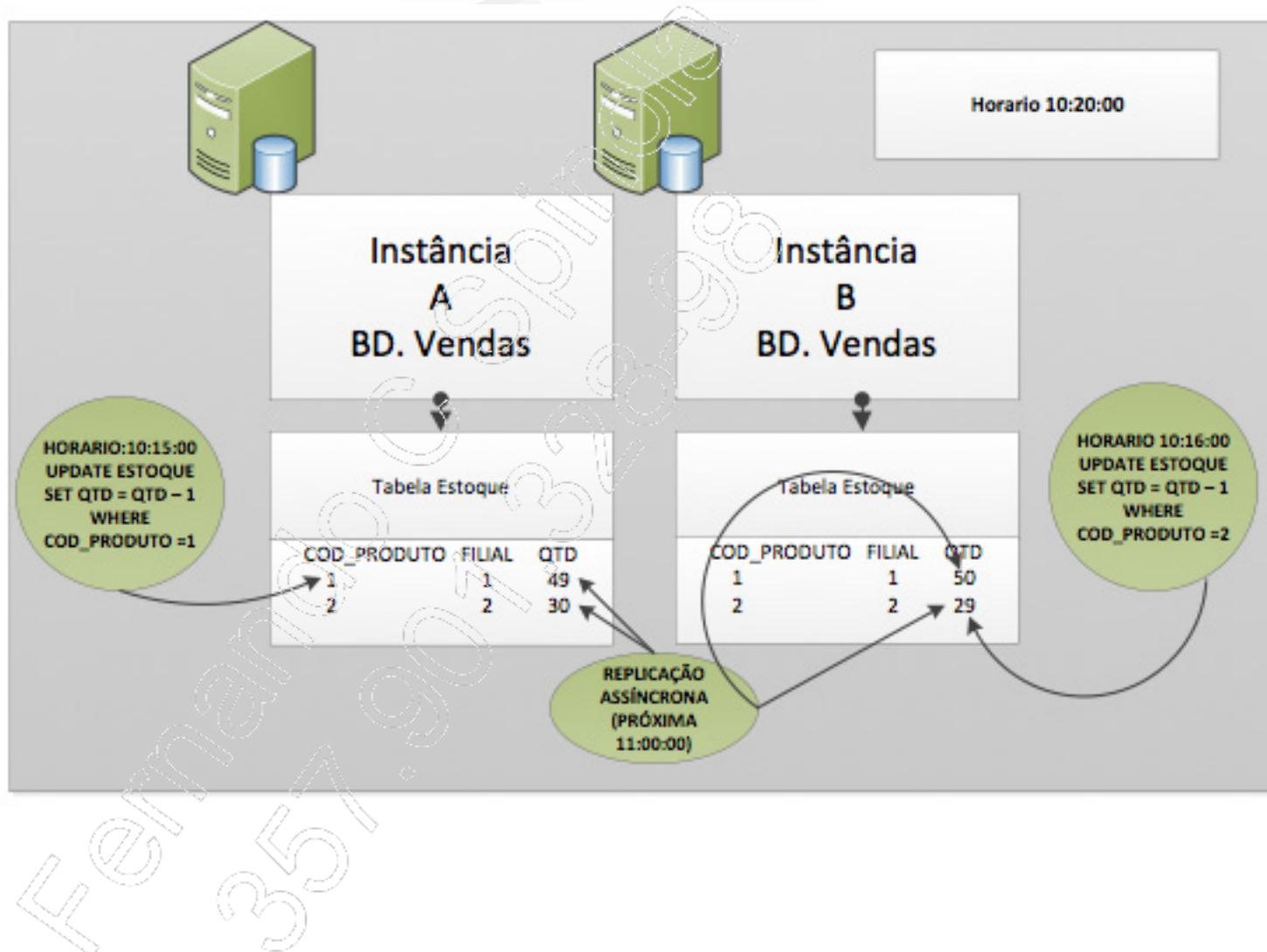
9.3.4. Assíncrona bidirecional

A replicação assíncrona bidirecional pode ser feita através da cópia dos dados modificados em um banco de dados (principal) para outro (réplica). Nesse segundo banco de dados, os dados podem sofrer manutenção (INSERT, UPDATE, DELETE ou MERGE), além de consultas. Este tipo de replicação é útil para manter os dados atualizados, porém pode apresentar latência (demora no envio de dados), pois é realizada em períodos de tempo estabelecidos, por exemplo, a cada hora. Neste caso, a replicação necessariamente adiará a atualização dos dados para os horários estabelecidos. Um exemplo deste tipo de replicação poderia ser o estoque do exemplo utilizado na replicação síncrona bidirecional.

SQL 2014 - Módulo III

Este tipo de replicação possui um problema que não encontramos nos demais que é o conflito de replicação, já que os dados são atualizados tanto no servidor principal quanto na réplica. Dessa forma, existe a necessidade de as ferramentas de replicação implementarem uma resolução para os eventuais conflitos de replicação que possam ocorrer.

A seguir, veremos um esquema de uma replicação assíncrona bidirecional:



9.4. Escolhendo a estratégia para deposição de dados

Alguns fatos devem ser considerados ao escolhermos o método de deposição de dados a ser utilizado no sistema. Os dois fatos principais são tempo de latência e autonomia de sites. O tempo de latência refere-se ao período em que pode não haver sincronia entre os dados presentes nos servidores que participam do processo de replicação.

O tempo de latência não pode ocorrer nos casos em que é necessária a atualização das cópias dos dados em todos os servidores nos quais elas estão contidas. Quando isso ocorre, o método de deposição de dados a ser utilizado deve ser a transação distribuída. Já quando pode haver certo tempo de latência, o método de deposição a ser utilizado é a replicação assíncrona.

O outro fato importante para a escolha adequada do método de deposição de dados é a autonomia de sites, que se refere ao nível de independência que um site tem em relação a outro servidor, ou seja, quanto tempo o servidor que recebeu os dados pode trabalhar sem precisar se reconectar ao servidor de origem desses dados a fim de obter uma nova cópia deles.

Quando utilizamos a replicação do tipo **Merge**, os servidores trabalham independentemente um do outro. Já quando trabalhamos com o protocolo **Two-phase commit**, cuja finalidade é controlar as transações entre os servidores, a transação apenas é processada se todos os servidores estiverem disponíveis. Caso contrário, tal processamento não é realizado em qualquer um desses servidores.

9.5. Replicação de dados no SQL Server

O processo de replicação do SQL Server realiza o envio de dados de um banco de dados de origem para um banco de dados de destino de forma periódica.

A fim de realizar esse processo, o SQL Server utiliza a seguinte metáfora de replicação: Editor (Editor/Publisher), Distribuidor (Distributor) e Assinante (Subscriber).

9.5.1. Metáfora da replicação

Implementar o processo de replicação é uma tarefa que requer a configuração dos servidores nele envolvidos. A metáfora de replicação já mencionada é um exemplo dessa configuração:

- **Editor (Editor/Publisher)**

É o servidor no qual os dados são originados. Este tipo de servidor possui o banco de dados de origem e publica os dados disponíveis para replicação enviando as alterações feitas nos dados que podem ser publicados no banco de dados Distributor. Este último pode estar presente no mesmo servidor em que o Publisher está localizado ou em um servidor distinto. Vale destacar que é mais comum encontrar o Distributor em um servidor distinto.

- **Distribuidor (Distributor)**

É o servidor responsável por distribuir os dados publicáveis. Ele recebe as alterações feitas nos dados que podem ser publicados, armazena essas alterações e as envia ao servidor Subscriber no intervalo de tempo determinado. Um servidor Distributor é capaz de suportar diversos servidores Publisher.

- **Assinante (Subscriber)**

É o servidor que receberá os dados enviados pelo servidor Distributor e as alterações feitas pelo servidor Publisher.

9.5.2. Publicações e artigos

Quando trabalhamos com a metáfora **Publisher/Distributor/Subscriber**, os dados replicados são referenciados como publicações (**Publications**) que possuem um ou mais artigos (**Articles**). Uma publicação deve ser criada a fim de permitir o envio de dados entre servidores por meio da replicação. A publicação pode conter um ou mais artigos, os quais representam um conjunto de dados a ser replicado.

Em um processo de replicação, um artigo pode ser simplesmente parte de uma publicação, assim como pode ser uma tabela ou um subconjunto de seus dados. Devemos ter em mente que, embora não possamos assinar um único artigo diretamente, podemos assinar a publicação que o contém.

9.5.2.1. Filtrando dados

Quando criamos uma publicação, podemos filtrar os dados que farão parte de um artigo. Essa filtragem permite publicar somente parte de uma tabela. Dessa forma, podemos escolher somente as colunas a serem replicadas, isto é, uma filtragem vertical de tabelas, ou podemos escolher somente as linhas a serem replicadas, isto é, uma filtragem horizontal. É possível, também, fazer a filtragem das duas formas, mas devemos ter em mente que cada instância da tabela filtrada representa um artigo separado.

Vejamos a seguir mais detalhes a respeito desses tipos de filtragem de dados:

- **Filtro vertical:** Este tipo de filtro possui um subconjunto de colunas de uma tabela. Quando utilizamos o filtro vertical, somente as colunas selecionadas são enviadas ao Subscriber;
- **Filtro horizontal:** Este tipo de filtro possui um subconjunto de linhas de uma tabela. Quando utilizamos o filtro horizontal, somente as linhas selecionadas são enviadas ao Subscriber;
- **Filtro horizontal/vertical:** Este tipo de filtro possui um subconjunto de colunas e linhas de uma tabela. Quando utilizamos o filtro horizontal/vertical, somente as linhas e colunas selecionadas são enviadas ao Subscriber.

9.6. Tipos de assinaturas

Os tipos de assinaturas que podemos utilizar são a **Push Subscription** e a **Pull Subscription**. A assinatura do tipo Push é configurada a partir do servidor Publisher e consome os recursos do servidor Distributor quando os dados são enviados ao Subscriber. A Push Subscription centraliza o processo de administração das assinaturas porque é definida no servidor Publisher e permite a configuração de diversos Subscribers de uma só vez para cada publicação.

 Uma assinatura pode ser configurada simultaneamente ao processo de criação ou de edição de uma publicação no servidor Publisher.

Já a assinatura Pull Subscription é configurada a partir de um servidor Subscriber e consome os recursos desse mesmo servidor. O assinante (Subscriber) também é responsável por iniciar esse tipo de assinatura. Devemos ter em mente, no entanto, que uma Pull Subscription somente pode ser feita por um servidor **Subscriber SQL Server**.

Uma assinatura Pull Subscription poderá ser configurada pelo administrador do sistema (system administrator – SA), pelos usuários pertencentes ao grupo Administrators do Windows ou pelo usuário proprietário do banco de dados em questão, no servidor Subscriber. É possível também registrar o Subscriber no servidor Publisher e fazer a assinatura do tipo Pull a partir dessa máquina.

9.7. Agentes de replicação

Os agentes da replicação que podem ser utilizados serão descritos na tabela a seguir:

Agentes de replicação	Descrição
Snapshot Agent	Este agente é utilizado com a finalidade de iniciar a sincronização dos dados das tabelas de origem e de destino dos dados, as quais fazem parte do processo de replicação. A função do Snapshot Agent é preparar arquivos snapshot dessas tabelas e armazená-las em um servidor do tipo Distributor.
Distribution Agent	Este agente tem a função de mover as transações ou os snapshots de dados do Publisher para o Subscriber. Durante o processo de distribuição dos dados nas tabelas de destino do banco de dados Subscription, o Distribution Agent monitora o processo de cópia da publicação.
Log Reader Agent	Este agente realiza o monitoramento do transaction log de cada banco de dados configurado para a publicação transacional, procurando por transações que devem ser replicadas. Ele faz uma cópia das transações do transaction log do Publisher que são marcadas para replicação. Feita a cópia, o agente log reader envia tais transações para o banco de dados Distribution, local em que elas são mantidas até que sejam distribuídas aos assinantes (subscribers).
Merge Agent	Este agente é responsável por fundir as alterações de dados realizadas em diversos locais desde a criação do primeiro snapshot. É utilizado com replicações do tipo Merge.

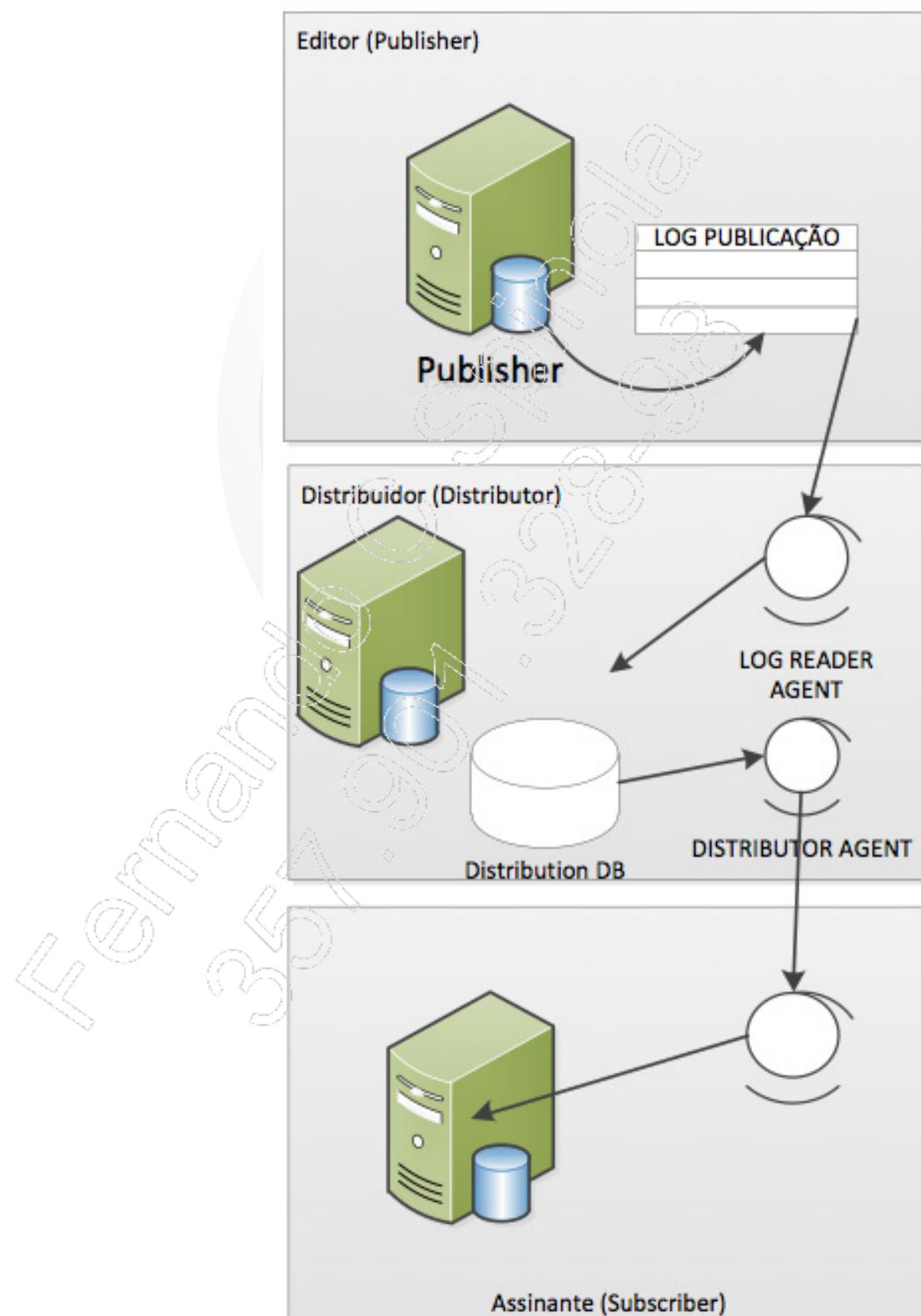


O snapshot Agent realiza a cópia da tabela tomando como base o BCP (Bulk Copy Program). O BCP não utiliza o log.

9.8. Tipos de publicação

O SQL Server permite criar tipos distintos de publicação, como **Snapshot**, **Transactional** e **Merge**. Estes três tipos de publicação serão abordados em detalhes na sequência deste capítulo. Vale destacar que podemos utilizar diversos tipos de publicação dentro dos bancos de dados.

A seguir, veremos uma figura que ilustra a replicação:



9.8.1. Snapshot Publication

Este tipo de publicação faz uma cópia, como se tirasse uma foto, dos dados publicados no banco de dados em um determinado momento. Por não fazer um monitoramento constante das alterações que são realizadas sobre os dados presentes no servidor de origem, a publicação Snapshot necessita de uma pequena quantidade de recursos do processador.

A publicação Snapshot envia ao assinante não apenas as alterações, mas todos os dados, fazendo com que os Subscribers sejam alterados por uma atualização (refresh) total do conjunto de dados replicados. Uma grande quantidade de recursos pode ser necessária para transmitir os dados, caso o artigo publicado seja muito grande.

Com a publicação Snapshot, realiza-se o tipo mais simples de replicação, o qual assegura a consistência entre o Publisher e o Subscriber e um alto índice de autonomia quando os dados não são alterados pelo Subscriber.

Este tipo de publicação é bastante utilizado nas seguintes situações: quando é necessário somente observar os dados e quando os dados são necessários apenas como um suporte à tomada de decisões. Por isso, trata-se de uma boa solução para os Subscribers read-only (somente leitura), que não necessitam dos dados mais atualizados.

9.8.2. Transactional Publication

Este tipo de publicação permite selecionar uma tabela inteira ou apenas parte dela, bem como permite selecionar uma ou mais stored procedures a fim de que elas sejam replicadas como um artigo dentro de uma publicação. A publicação Transactional utiliza o transaction log com a finalidade de capturar as alterações feitas sobre os dados que pertencem aos artigos marcados para replicação.

O SQL Server realiza o monitoramento dos comandos INSERT, UPDATE e DELETE que são executados sobre os dados marcados para replicação e armazena tais alterações no banco de dados de distribuição do servidor Distribution. Feito isto, as alterações são enviadas aos Subscribers e aplicadas na mesma ordem.

A publicação Transactional envia as alterações feitas no Publisher ao Subscriber de forma contínua ou de acordo com os intervalos previamente definidos. Quando trabalhamos com este tipo de publicação, os conflitos são evitados, uma vez que a alteração dos dados é feita somente no servidor responsável por publicá-los. Ao evitar conflitos, a publicação Transactional assegura a consistência da transação, pois todos os servidores assinantes terão os mesmos valores que o editor no qual as atualizações foram realizadas.

Caso os Subscribers necessitem receber os dados praticamente em tempo real, eles precisam estar constantemente conectados ao Publisher em servidores com publicações transacionais. Também é possível que o Subscriber obtenha as alterações feitas sobre os dados apenas no momento em que isso for necessário. Essa publicação representa uma opção adequada para os usuários desconectados que desejam acessar dados read-only (somente leitura).

9.8.3. Merge Publication

Este tipo de publicação permite que todos os sites envolvidos no processo tenham todos os dados. No entanto, a publicação Merge pode gerar conflitos, uma vez que as alterações dos dados feitas no banco de dados de destino são propagadas para o banco de dados de origem. O servidor que criar a publicação será o servidor Publisher.

Quando a Merge Publication é utilizada para publicar uma tabela, o esquema do banco de dados sofre três importantes alterações, a destacar:

- Para que cada linha da tabela seja identificada como única, uma coluna do tipo unique identifier é incluída pelo SQL Server na tabela a ser publicada. Caso a tabela já possua este tipo de coluna definido com a propriedade ROWGUIDCOL, a coluna em questão é automaticamente utilizada pelo SQL Server para identificar cada linha. Além disso, o SQL Server inclui um índice na coluna **rowguid** da tabela. Ambos, coluna e índice, são adicionados à tabela quando o Snapshot Agent executa a publicação pela primeira vez ou quando o artigo é ativado em tempo de criação;

- Triggers que procuram por alterações feitas sobre os dados da tabela publicada são criados pelo SQL Server. As alterações capturadas pelos triggers são registradas na tabela do sistema. Vale destacar que o SQL Server é capaz de suportar diversos triggers do mesmo tipo em uma tabela, por isso, os triggers gerados pelas publicações Merge não interferem nos triggers definidos pelas aplicações;
- Diversas tabelas do sistema são adicionadas pelo SQL Server no banco de dados de publicação.

9.8.4. Resolução de conflitos

Tendo em vista que a publicação Merge permite a alteração dos dados por parte do Publisher e dos Subscribers, é possível que ocorram conflitos. O Merge Agent obtém informações da tabela do sistema **Msmerge_contents** a fim de detectar esses conflitos. Assim que detecta algum conflito, ele verifica qual alteração deve ser atribuída aos dados publicados, aplicando-a em seguida.

9.9. Cenário de replicação

Contamos com os cenários de replicação nos quais os dados são replicados entre servidores e com os cenários nos quais os dados são replicados entre cliente e servidor. O tipo de replicação utilizado no primeiro cenário é Transactional, mas também é possível utilizar a replicação Snapshot. Já no segundo cenário mencionado, no qual o processo de replicação é realizado entre cliente e servidor, o tipo de replicação utilizado é Merge.

9.9.1. Cenário de replicação cliente/servidor

A replicação de dados entre cliente e servidor é uma tarefa normalmente realizada com a finalidade de oferecer suporte a aplicações que realizam a troca de dados com usuários móveis, a aplicações que realizam a troca de dados provenientes de diversos sites e a aplicações POS (Point of Sale).

CRM (Customer Relationship Management), SFA (Sales Force Automation) e FFA (Field Force Automation) são exemplos de aplicações que necessitam trocar dados com usuários remotos. Já as aplicações POS são do tipo que necessita da replicação dos dados provenientes de diversos sites remotos para um site centralizado.

9.9.2. Cenário de replicação entre servidores

Os cenários de replicação entre servidores são normalmente utilizados de acordo com as finalidades descritas a seguir:

- **Integração de dados:** Os cenários que permitem o processo de replicação entre servidores são utilizados com a finalidade de permitir a integração de dados provenientes de diversos sites, bem como a integração de dados enviados de ou para bancos de dados que não pertencem ao SQL Server;
- **Melhor disponibilidade e escalabilidade:** Os cenários nos quais o processo de replicação é realizado entre servidores permitem manter cópias de dados sempre atualizadas. Isso oferece maior escalabilidade entre os servidores com relação à leitura desses dados e impede a redundância gerada pela manutenção de diversas cópias dos mesmos dados;
- **Dados presentes em servidores OLTP:** Os cenários de replicação entre servidores permitem mover os dados não apenas entre servidores OLTP, mas também entre sistemas de suporte à decisão e aos relatórios;
- **Servidor dedicado de processamento em lote:** Tendo em vista que as operações em lote costumam necessitar de uma grande quantidade de recursos para serem executadas em servidores OLTP, devemos utilizar a replicação para reduzir a carga incidente sobre um servidor de processamento em lote dedicado.

9.10. Restrições de replicação

Antes que possamos configurar um processo de replicação, devemos conhecer as restrições aplicadas sobre esse processo: cada uma das publicações pode conter artigos provenientes de apenas um banco de dados; as tabelas que fazem parte da configuração de uma publicação Transactional devem conter chave primária; uma coluna do tipo unique identifier é adicionada pelo SQL Server na tabela durante a configuração de uma publicação do tipo **Merge**; os dados dos seguintes bancos de dados de sistema não podem ser replicados: Master, Model, Distribution, MSDB e TempDB.



Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo.

- Um ambiente de distribuição de dados é aquele que pode conter cópias dos mesmos dados em diversos servidores. A distribuição de dados pode ser implementada por meio de duas estratégias principais: replicação e transação distribuída;
- O processo de replicação do SQL Server realiza o envio de dados de um banco de dados de origem para um banco de dados de destino de forma periódica. A fim de realizar esse processo, o SQL Server utiliza a seguinte metáfora de replicação: Publisher (Editor), Distributor (Distribuidor) e Subscriber (Assinante);
- Contamos com os cenários de replicação nos quais os dados são replicados entre servidores (replicação Transactional ou Snapshot) e com os cenários nos quais os dados são replicados entre cliente e servidor (replicação Merge);
- Antes que possamos configurar um processo de replicação, devemos conhecer as restrições aplicadas sobre esse processo: cada uma das publicações pode conter artigos provenientes de apenas um banco de dados; as tabelas que fazem parte da configuração de uma publicação Transactional devem conter chave primária; uma coluna do tipo unique identifier é adicionada pelo SQL Server na tabela durante a configuração de uma publicação do tipo Merge; os dados dos seguintes bancos de dados de sistema não podem ser replicados: Master, Model, Distribution, MSDB e TempDB.

9

Replicação e distribuição de dados

Teste seus conhecimentos

Fernando Góspodola
357.907.



IMPACTA
EDITORA

1. Sobre tipos de replicação, qual alternativa está errada?

- a) Síncrona unidirecional
- b) Síncrona bidirecional
- c) Síncrona multidirecional
- d) Assíncrona unidirecional
- e) Assíncrona bidirecional

2. Qual(is) é(são) a(s) estratégia(s) de deposição de dados?

- a) Replicação.
- b) Distribuição de dados.
- c) Cluster.
- d) Replicação e cópia de dados.
- e) Replicação e distribuição de dados.

3. Qual é o protocolo de banco de dados utilizado em replicação e distribuição?

- a) Two-phase commit
- b) Two-phase sync
- c) Two-phase transaction log
- d) Transaction Log
- e) Merge Agent

4. Qual alternativa não é um tipo de publicação?

- a) Snapshot
- b) Transactional
- c) Log
- d) Merge
- e) Nenhuma das alternativas anteriores.

5. O que determina o uso da replicação assíncrona ao invés da síncrona?

- a) O tempo de latência e a autonomia dos sites.
- b) O modelo síncrono sempre deve ser utilizado.
- c) Os dois tipos realizam réplicas.
- d) Não há diferença entre os tipos.
- e) Assíncrona é mais rápida do que síncrona.

9

Replicação e distribuição de dados

Mãos à obra!

Fernando
357.90
Sócio-inovador



IMPACTA
EDITORA

Laboratório 1

Este laboratório será feito em dupla. Vamos configurar uma publicação do tipo **Merge**. Por isso, é fundamental que dois alunos executem juntos cada passo deste exercício.

A – Configurando uma replicação

1. Escreva o nome do servidor **Publisher/Distributor**: _____
2. Escreva o nome do servidor **Subscriber**: _____
3. Sua dupla e você devem abrir o Microsoft SQL Server Management Studio. Para tanto, cada um clique no botão **Start**, em seguida abra a opção **All Programs** e depois escolha o **Microsoft SQL Server 2014**. Escolha **SQL Server Management Studio** e conecte-se com a autenticação do Windows;
4. Para configurar esta replicação, mantenha a conexão dos dois servidores (**Publisher** e **Subscriber**) registrada no **Object Explorer**. Para fazer esta conexão, sua dupla e você deverão seguir os passos descritos adiante:
 - 4.1. No menu de ferramentas do **Object Explorer**, clique na opção **Connect** e selecione **Database Engine**;
 - 4.2. Na tela que será exibida, no campo **Server Name**, escolha o nome do servidor da sua dupla;
 - 4.3. No campo **Authentication**, escolha a opção **Windows Authentication**;
 - 4.4. Clique no botão **OK**;
 - 4.5. Observe que agora existem duas conexões no **Object Explorer**: uma no seu servidor e outra no servidor da sua dupla.

5. Configure o servidor **Distribuidor**. Para isso, a pessoa que está no servidor escolhido como **Publisher/Distributor** deverá realizar os passos a seguir:

5.1. Prepare o ambiente para realizar a replicação:

5.1.1. Clique sobre o nome do seu servidor, deixando-o selecionado;

5.1.2. Na barra de ferramentas, escolha a opção **New Query**;

5.1.3. Na barra de menus, escolha a opção **File**, em seguida, escolha **Open** e depois **File** novamente;

5.1.4. Abra o **script_01_Merge** da pasta **Capitulo_09**, conectando-se com **Windows Authentication**;

5.1.5. Esse script cria o database chamado **BancoReplica**. Crie dentro desse banco a tabela chamada **Treinamento** e insira nela seis registros. Para executar este script, pressione F5.

5.2. Na sequência, configure o mesmo servidor em que foi executado o script anterior como o **Distributor**. Para tanto, siga estes passos:

5.2.1. Clique com o botão direito do mouse sobre a pasta **Replication**;

5.2.2. Escolha a opção **Configure Distribution**;

5.2.3. Clique no botão **Next**;

5.2.4. Na tela que será exibida, assegure que esteja selecionado o nome do servidor correto para ser configurado como nosso **Distributor**. Aceite essa opção como default e clique em **Next**;

5.2.5. Aceite a pasta default como o **Snapshot Folder** e clique em **Next**;

- 5.2.6. Aceite as opções escolhidas como default para a criação do database **Distribution** e clique em **Next**;
 - 5.2.7. A próxima tela habilita o servidor selecionado para utilizar esse distribuidor quando for publicar algum dado. Aceite o default e clique na opção **Next**;
 - 5.2.8. Clique em **Finish**, verifique as opções que serão configuradas e, depois, clique em **Finish** novamente;
 - 5.2.9. Observe o SQL Server realizando as configurações estabelecidas;
 - 5.2.10. Assegure-se de que os passos anteriores foram realizados com sucesso;
 - 5.2.11. Clique no botão **Close**.
- 5.3. Verifique a criação do database **Distribution** neste servidor. Para tanto, siga estes passos:
 - 5.3.1. No **Object Explorer**, expanda a pasta **Databases**;
 - 5.3.2. Expand a pasta **System Databases**;
 - 5.3.3. Observar que, na lista dos databases do sistema, agora também existe um database chamado **Distribution**;
 - 5.3.4. Feche a pasta **System Databases**;
 - 5.3.5. Feche a pasta **Databases**.
6. Configure a publicação, ainda no mesmo servidor que foi configurado como **Distributor**. Para tanto, siga estes passos:
 - 6.1. Expand a pasta **Replication**;

- 6.2. Clique com o botão direito do mouse sobre a pasta **Local Publications**;
- 6.3. Escolha a opção **New Publication**;
- 6.4. Na tela que será exibida, clique no botão **Next**;
- 6.5. Na próxima tela, selecione o nome do database que terá os dados publicados. Neste caso, escolha **BancoReplica**;
- 6.6. Clique no botão **Next**;
- 6.7. Na próxima tela, escolha o tipo de publicação a ser configurada. Neste caso, selecione **Merge Publication**;
- 6.8. Clique no botão **Next**;
- 6.9. Na próxima tela, aceite como default a opção marcada **SQL Server 2014** e clique no botão **Next**;
- 6.10. Na próxima tela, expanda a opção **Tables** e selecione a tabela **Treinamento**;
- 6.11. Clique no botão **Next**;
- 6.12. A próxima tela informa que será acrescentada uma coluna na tabela **Treinamento**, cujo datatype será **Uniqueidentifier**. Clique, então, no botão **Next**;
- 6.13. Clique em **Next** novamente;
- 6.14. Na próxima tela, não será necessário adicionar nenhum filtro. Clique em **Next**;
- 6.15. A próxima tela especifica quando o **Snapshot Agent** deve rodar. Aceite o default e clique em **Next**;

SQL 2014 - Módulo III

- 6.16. Na próxima tela, clique no botão **Security Settings**;
 - 6.17. Na tela que será exibida, escolha a opção **Run under the SQL Server Agent Service Account** e clique na opção **OK**;
 - 6.18. Clique no botão **Next**;
 - 6.19. Na próxima tela, aceite o default e clique em **Next**;
 - 6.20. Na tela seguinte, atribua um nome para esta publicação. Para tanto, no campo **Publication name** escreva **PublicaçãoMerge**;
 - 6.21. Clique no botão **Finish**;
 - 6.22. Observe o **SQL Server** executando as configurações estabelecidas;
 - 6.23. Assegure-se de que todas as configurações tenham sido executadas com sucesso;
 - 6.24. Clique no botão **Close**;
 - 6.25. No **Object Explorer**, observe na pasta **Local Publications** que a publicação atual faz parte da lista de publicações deste servidor;
 - 6.26. Expanda **SQL Server Agent** e a pasta **Jobs**. Note que os jobs de replicação foram todos criados pelo processo de configuração realizado até aqui.
7. No servidor **assinante (do outro aluno)**, prepare o ambiente para realizar a assinatura da publicação. Para tanto, observe os seguintes passos:
 - 7.1. Clique sobre o nome do servidor **Assinante** e, na barra de ferramentas, clique no botão **New Query**;
 - 7.2. Conecte-se com a autenticação do Windows;

7.3. Escreva e execute o comando a seguir:

```
CREATE DATABASE BancoReplica  
go
```

8. Ainda no servidor **Assinante**, configure a assinatura da publicação. Para tanto, siga estes passos:

- 8.1. No servidor assinante, expanda a pasta **Replication**;
- 8.2. Clique com o botão direito do mouse sobre a pasta **Local Subscriptions** e escolha a opção **New Subscriptions**;
- 8.3. Clique no botão **Next**;
- 8.4. Na tela que será exibida, no campo **Publisher**, escolha a opção **Find SQL Server Publisher**;
- 8.5. Surgirá a tela para que seja feita a conexão com o servidor que possui a publicação. Conecte-se a ele com a autenticação do Windows;
- 8.6. Observe que a próxima tela exibe o nome das publicações disponíveis para serem assinadas. Selecione a **PublicaçãoMerge** e clique no botão **Next**;
- 8.7. Na tela que será exibida, selecione a opção **Run all agents at the Distributor**;
- 8.8. Clique no botão **Next**;
- 8.9. Na tela que será exibida, no campo **Subscription Database**, escolha o nome do database que receberá os dados replicados. Neste caso, escolha **BancoReplica** e clique no botão **Next**;
- 8.10. Na próxima tela, no campo **Connection to subscriber**, clique nas reticências (...);

- 8.11. Na tela que será exibida, escolha a opção **Run under the SQL Server Agent service account...**;
 - 8.12. Aceite as demais opções como default e clique no botão **OK**;
 - 8.13. Clique em **Next**;
 - 8.14. Na próxima tela (**Synchronization Schedule**), observe que no campo **Schedule** está marcada a opção **Run Continuously**. Aceite o default e clique em **Next**;
 - 8.15. Na próxima tela (**Initialize Subscription**), aceite o default e clique em **Next**;
 - 8.16. Na tela **Subscription Type**, aceite o default e clique em **Next**;
 - 8.17. Na tela seguinte (**Wizard Action**), aceite o default e clique em **Next**;
 - 8.18. Clique no botão **Finish**;
 - 8.19. Observe o SQL Server executando as configurações;
 - 8.20. Assegure-se de que tudo ocorreu com sucesso e clique no botão **Close**.
-
9. Agora, ainda no servidor **Assinante**, verifique se a tabela **Treinamento** foi criada e se ela já recebeu os dados publicados. Para isso, siga os passos adiante:
 - 9.1. No servidor **Assinante**, na barra de ferramentas, escolha a opção **New Query** e abra o **script_02_Merge** que acessa o database **BancoReplica** e realiza uma leitura de dados da tabela **Treinamento**;
 - 9.2. Com o script aberto, pressione F5 para executá-lo;
 - 9.3. Observe que a tabela **Treinamento** já foi criada nesta máquina e que já possui dados;

Replicação e distribuição de dados

9.4. Neste servidor, escreva e execute os comandos a seguir:

```
Use BancoReplica  
go  
INSERT Treinamento(Cod_Trein, Nome_Trein) VALUES(7, 'Interbase')  
INSERT Treinamento(Cod_Trein, Nome_Trein) VALUES(8, 'Miracle')
```

10. Agora, na máquina do servidor **Publisher (Editor)**, verifique se os dados inseridos no **Subscriber** já chegaram no **Publisher**. Para tanto, siga estes passos:

- 10.1. Expanda a pasta **Databases** e o database **BancoReplica**;
- 10.2. Expanda a pasta **Databases**;
- 10.3. Clique com o botão direito do mouse sobre a tabela **Treinamento** e escolha a opção **Select TOP 1000 Rows**;
- 10.4. Observe que os dois registros inseridos pelo servidor assinante já foram replicados para o **Publisher**;
- 10.5. Feche esta tabela.

11. Insira mais dois registros na tabela **Treinamento** a partir do servidor **Publisher**. Para tanto, siga este passo a passo:

- 11.1. No **Object Explorer**, clique sobre o nome do servidor **Publisher**;
- 11.2. No servidor **Publisher**, na barra de ferramentas, escolha a opção **New Query**. Conecte-se com a autenticação do Windows;
- 11.3. Escreva e execute os comandos a seguir (**Script_03**):

```
USE BancoReplica  
go  
INSERT Treinamento(Cod_Trein, Nome_Trein) VALUES(9, 'Cobol1')  
INSERT Treinamento(Cod_Trein, Nome_Trein) VALUES(10, 'Cobol2')  
go  
SELECT * FROM Treinamento
```

11.4. Note que esta tabela agora possui dez registros;

11.5. Clique sobre o nome do servidor **Assinante** e, na barra de ferramentas, escolha a opção **New Query**. Conecte-se com a conta do Windows;

11.6. Escreva e execute os comandos a seguir:

```
USE BancoReplica  
go  
SELECT * FROM Treinamento  
Go
```

11.7. Observe que, nesta máquina, a tabela **Treinamento** também possui os dez registros inseridos;

11.8. Feche o SQL Server Management Studio sem salvar os scripts.

Laboratório 2

No exercício anterior, configuramos uma replicação, cuja publicação era do tipo **Merge**, entre dois servidores remotos. Neste laboratório, vamos realizar outra replicação, mas desta vez será entre duas instâncias locais. Para tanto, inicialmente vamos desconfigurar a replicação anterior e, em seguida, configurar a atual.

Neste instante, deve-se ter duas conexões no SQL Server Management: uma local (da nossa instância) e outra com a máquina remota. Vamos executar as etapas a seguir para configurar a replicação do tipo **Transactional**:

- Desconectar o servidor remoto;
- Conectar a segunda instância local;
- Configurar o distribuidor;
- Preparar o ambiente do servidor Publisher com os dados a serem replicados;

- Preparar o ambiente do servidor Subscriber;
- Criar a publicação Transactional;
- Assinar a publicação Transactional;
- Fazer as verificações.

A – Desconectando o servidor remoto

1. No SQL Server Management Studio, do lado esquerdo da tela, no **Object Explorer**, feche todas as pastas abertas nas duas instâncias até que apareçam apenas os nomes das duas instâncias conectadas;
2. Clique com o botão direito do mouse sobre a instância remota (do colega) e escolha a opção **Disconnect**.

B – Conectando a segunda instância local

Vamos agora fazer uma conexão com a segunda instância local:

1. No menu de ferramentas do **Object Explorer**, clique na opção **Connect** e escolha a opção **Database Engine**;
2. Na tela que será exibida, no campo **Server Name**, escolha o nome da segunda instância;
3. No campo **Authentication**, escolha a opção **Windows Authentication**;
4. Clique no botão **OK**;
5. Observe que, agora, no **Object Explorer**, existem duas conexões: uma na primeira instância e outra na segunda instância;
6. Deixe as duas instâncias fechadas (minimizadas).

C – Configurando o distribuidor

Agora, vamos configurar o servidor **Distribuidor** na primeira instância. Para tanto, devemos seguir os passos adiante:

1. Clique sobre a primeira instância;
2. Expanda a primeira instância;
3. Clique com o botão direito do mouse sobre a pasta **Replication** e escolha a opção **Configure Distribution**;
4. Na tela que será exibida, clique no botão **Next**;
5. A próxima tela indica que o **Distribuidor** será configurado nesta instância. Portanto, aceite o default e clique em **Next**;
6. Aceite o default na próxima tela e clique em **Next**;
7. Observe o nome do database **Distribution** e o local de seus arquivos. Aceite as opções como default e clique em **Next**;
8. Observe que, na próxima tela, a primeira instância local está sendo habilitada a publicar dados. Aceite o default e clique em **Next**;
9. Na próxima tela está selecionada a opção **Configure distribution**. Aceite o default e clique em **Next**;
10. Na tela seguinte, observe um relatório das configurações que serão feitas. Clique em **Finish**;
11. Observe o SQL Server realizando as configurações determinadas;
12. Assim que tudo for feito com sucesso, clique em **Close**.

D – Preparando o ambiente do servidor Publisher com os dados a serem replicados

Neste momento, no **Object Explorer**, deve-se ter a segunda instância minimizada e a primeira instância expandida. Ainda na primeira instância, vamos preparar os dados que serão replicados:

1. Na barra de menus, clique em **File**, em seguida em **Open** e depois em **File** novamente;
2. Na pasta **Capitulo_09**, dê um duplo-clique sobre o **script_04_Transact_Um**;
3. Surgirá a tela para a escolha do tipo de conexão. Escolha **Conexão do Windows**. Observe que, no campo **Server name**, deverá estar marcado o nome da primeira instância. Clique no botão **Connect**;
4. Com o **script_04_Transact_Um** aberto, pressione F5 para executá-lo;
5. Feche a query desse script;
6. Do lado esquerdo da tela, no **Object Explorer**, feche todas as pastas que tenham sido abertas e feche (minimize) a instância.

E – Preparando o ambiente do servidor Subscriber

Neste momento, devemos ter as duas instâncias fechadas (minimizadas) no **Object Explorer**.

Agora, vamos preparar o ambiente na segunda instância para que ela possa receber os dados que serão replicados pela primeira instância:

1. Clique sobre a segunda instância;
2. Expanda a segunda instância;
3. Na barra de ferramentas, clique na opção **File**, em seguida, em **Open** e depois em **File** novamente;

4. Encontre o **script_05_Transact_Um** da pasta **Capítulo_09** e efetue um duplo-clique sobre ele;
5. Na tela que será exibida, no campo **Server name**, escolha o nome da segunda instância e conecte-se ao SQL Server com a autenticação do Windows;
6. Este script cria apenas o database para receber os dados replicados. Pressione F5 para executá-lo;
7. Feche a query do **script_05_Transact_Um**;
8. Do lado esquerdo da tela, no **Object Explorer**, feche todas as pastas que tenham sido abertas e feche (minimize) a segunda instância.

F – Criando a publicação Transactional

1. Do lado esquerdo da tela, no **Object Explorer**, expanda a primeira instância;
2. Expanda a pasta **Replication**;
3. Clique com o botão direito do mouse sobre a pasta **Local Publications** e escolha a opção **New Publication**;
4. Na tela que será exibida, clique no botão **Next**;
5. Selecione, na próxima tela, o database **BancoTransact_Um** e clique em **Next**;
6. Na próxima tela, você pode escolher o tipo de publicação. Selecione a opção **Transactional publication** e clique em **Next**;
7. Na tela que será exibida, expanda o ícone **Tables**, selecione a tabela **Material** e clique no botão **Next**;
8. Clique em **Next** na nova tela;

9. Na tela será exibida, selecione as opções **Create snapshot immediately and keep the snapshot available to initialize subscriptions** e **Schedule the Snapshot Agent to run at the following times**. Em seguida, clique no botão **Change**;
10. Na tela que será exibida, na seção **Daily frequency**, selecione a opção **Occurs every** e escolha **1 Minute(s)**;
11. Aceite todas as outras opções desta tela como default e clique no botão **Next**;
12. Clique em **Next** novamente;
13. Na próxima tela, clique no botão **Security Settings** e selecione a opção **Run under the SQL Server Agent service account....** Em seguida, clique no botão **OK**;
14. Na próxima tela, clique em **Next**;
15. Na tela seguinte, selecione a opção **Create the publication** e clique em **Next**;
16. Na nova tela, no campo **Publication name**, escreva **PublicacaoTransacional** e clique em **Finish**;
17. Observe o SQL Server realizar as configurações;
18. Assim que terminar com sucesso, clique em **Close**.

G – Assinando a publicação Transactional

1. Ainda na primeira instância, no servidor **Publisher**, observe do lado esquerdo da tela, no **Object Explorer**, que a publicação de nome **PublicacaoTransacional** já faz parte da lista de publicações da pasta **Local Publications**;
2. Clique com o botão direito do mouse sobre a publicação de nome **PublicacaoTransacional** e escolha a opção **New Subscriptions**;

SQL 2014 - Módulo III

3. Na tela que será exibida, selecione a publicação de nome **PublicacaoTransacional** e clique no botão **Next**;
4. Na próxima tela, aceite a primeira opção selecionada como default e clique em **Next**;
5. Na tela seguinte, clique no botão **Add Subscriber** e selecione a opção **Add SQL Server Subscriber...**;
6. Surgirá a tela de conexão. No campo **Server name**, selecione o nome da segunda instância, faça a conexão com a autenticação do Windows e clique em **Connect**;
7. Note que, na tela anterior, ficou selecionado apenas o nome da segunda instância. Aceite o default e clique em **Next**;
8. Na próxima tela, no campo **Connection to Subscriber**, clique no botão com as reticências (...);
9. Na tela que será exibida, selecione a opção **Run under the SQL Server Agent service account...** e clique no botão **OK**;
10. Clique em **Next**;
11. Note que, no campo **Agent Schedule**, está selecionada a opção **Run continuously**. Clique em **Next**;
12. Na próxima tela, note que, no campo **Initialize when**, está selecionada a opção **Immediately**. Clique em **Next**;
13. Na próxima tela, deixe selecionada a opção **Create Subscription(s)** e clique em **Next**;
14. Na tela seguinte, observe o relatório de tudo o que será feito pelo **SQL Server** e clique em **Finish**;
15. Observe o SQL Server executando todas as configurações;
16. Assim que tudo terminar com sucesso, clique em **Close**.

H – Executando as verificações

1. Expanda a segunda instância e a pasta **Databases**;
2. Clique com o botão direito do mouse sobre a pasta **Databases** e escolha a opção **Refresh**;
3. Expanda o database **BancoTransact_Um**;
4. Expanda a pasta **Tables** e note que a tabela **Material** já foi replicada para a segunda instância;
5. Clique com o botão direito do mouse sobre a tabela **Material** e escolha a opção **Open Table**;
6. Note que cinco materiais foram replicados para esta tabela, cujos dados foram enviados do servidor **Editor (Publisher)**;
7. Clique novamente sobre o nome do servidor da primeira instância e, no menu de opções, clique em **File / Open / File**;
8. Abra a pasta **Capitulo_09** e dê um duplo-clique sobre o **script_06_Transact_Um**;
9. Aparecerá a tela de conexão. No campo **Server name**, escolha o nome da primeira instância e, com a autenticação do **Windows**, clique no botão **Connect**;
10. O **script_06_Transact_Um** inserirá mais cinco registros na tabela **Material** da primeira instância. Pressione F5 para executá-lo;
11. Aguarde alguns instantes e, no **Object Explorer**, na segunda instância, clique com o botão direito do mouse sobre a tabela **Material**. Escolha a opção **Open Table** e, na barra de ferramentas, clique no ícone que representa um ponto de exclamação (!) para que a tabela seja lida novamente;
12. Note que os cinco registros novos já foram recebidos pelo servidor **Assinante**;

13. Ainda na segunda instância, na barra de menus, escolha a opção **File / Open** e, em seguida, **File** novamente. Aplique um duplo-clique sobre o **script_07_Transact_Um**;
14. Na tela que será exibida, no campo **Server name**, selecione o nome da segunda instância e conecte-se com a autenticação do **Windows**. Clique no botão **Connect**;
15. O **script_07_Transact_Um** insere na tabela **Material** do servidor **Assinante** mais cinco registros, que não serão enviados para o servidor **Publisher**, como acontece no caso da publicação do tipo **Merge**. Pressione F5 para executar esse script;
16. Aguarde alguns instantes para que os jobs de replicação sejam executados novamente;
17. Na primeira instância, expanda a pasta **Databases**;
18. Expanda o database **BancoTransact_Um** e a pasta **Tables**;
19. Clique com o botão direito do mouse sobre a tabela **Material** e escolha a opção **Open Table**. Note que os dados inseridos no servidor **Assinante (Subscriber)** não foram enviados para o servidor **Editor (Publisher)**;
20. Nas duas instâncias, feche todas as queries que foram abertas e também todas as pastas que tenham sido abertas;
21. Para encerrar, feche (minimize) as duas instâncias.

Laboratório 3

No exercício anterior, configuramos uma replicação, cuja publicação era do tipo **Transactional**, entre duas instâncias. Neste laboratório, vamos realizar uma replicação cuja **Publicação** será do tipo **Transactional “Ponto a Ponto”**. Utilizaremos duas instâncias locais novamente.

Replicação e distribuição de dados

Neste laboratório, utilizaremos o **Distribuidor** que já foi configurado anteriormente. Este exercício deverá ser executado através das seguintes etapas:

- Preparar o ambiente do servidor Publisher com os dados a serem replicados;
- Preparar o ambiente do servidor Subscriber;
- Criar a publicação Transactional Ponto a Ponto;
- Assinar a publicação Transactional;
- Executar as verificações.

A – Preparando o ambiente do servidor Publisher com os dados a serem replicados

Neste momento, no **Object Explorer**, devemos ter as duas instâncias minimizadas. Na primeira instância, vamos preparar os dados que serão replicados. Para tanto, devemos seguir estes passos:

1. Na barra de menus, clique em **File**, em seguida, em **Open** e depois em **File** novamente;
2. Na pasta **Capítulo_09**, efetue um duplo-clique sobre o **script_08_Transact_Dois**;
3. Surgirá a tela para a escolha do tipo de conexão. Escolha **Conexão do Windows**. Observe que, no campo **Server name**, deverá estar marcado o nome da primeira instância. Clique no botão **Connect**;
4. Com o **script_08_Transact_Dois** aberto, pressione F5 para executá-lo;
5. Feche a query desse script.

B – Preparando o ambiente do servidor Subscriber

Neste momento, devemos ter as duas instâncias fechadas (minimizadas) no **Object Explorer**. Agora, vamos preparar o ambiente na segunda instância para que ela possa receber os dados que serão replicados pela primeira instância. Para tanto, siga estes passos:

1. Na barra de ferramentas, clique na opção **File**, em seguida, em **Open** e depois em **File** novamente;
2. Encontre o **script_09_Transact_Dois** na pasta **Capitulo_09** e aplique um duplo-clique sobre ele;
3. Na tela que será exibida, no campo **Server name**, escolha o nome da segunda instância e conecte-se ao SQL Server com a autenticação do Windows;
4. Este script apenas cria o database para receber os dados replicados. Pressione F5 para executá-lo;
5. Feche a query desse script.

C – Criando a publicação Transactional

1. Do lado esquerdo da tela, no **Object Explorer**, expanda a primeira instância;
2. Expanda a pasta **Replication**;
3. Clique com o botão direito do mouse sobre a pasta **Local Publications** e escolha a opção **New Publication**;
4. Na tela que será exibida, clique no botão **Next**;
5. Na próxima tela, selecione o database **BancoTransact_Dois** e clique em **Next**;
6. Na próxima tela, você pode escolher o tipo de publicação. Neste exercício, você deve escolher a terceira opção. Portanto, selecione a opção **Transactional publication with updatable subscriptions** e clique em **Next**;

7. Na tela que será exibida, expanda o ícone **Tables** e selecione a tabela **Material**;
8. Na próxima tela, clique em **Next**;
9. A tela seguinte indica que será acrescentada na tabela **Material** uma coluna do tipo **uniqueidentifier**. Clique em **Next**;
10. Na próxima tela, clique em **Next** novamente;
11. Na nova tela, selecione as opções **Create snapshot immediately and keep the snapshot available to initialize subscriptions** e **Schedule the Snapshot Agent to run at the following times**. Em seguida, clique no botão **Change**;
12. Na tela que será exibida, na seção **Daily frequence**, selecione a opção **Occurs every**, escolha **1 Minute(s)** e clique em **OK**;
13. Aceite todas as outras opções desta tela como default e clique no botão **Next**;
14. Clique em **Next** novamente;
15. Na próxima tela, clique no botão **Security Settings** e selecione a opção **Run under the SQL Server Agent service account**. Em seguida, clique no botão **OK**;
16. Clique em **Next**;
17. Na tela seguinte, selecione a opção **Create the publication** e clique em **Next**;
18. Na próxima tela, no campo **Publication name**, escreva **PublicacaoTransacional_Ponto_A_Ponto** e clique em **Finish**;
19. Observe o SQL Server realizar as configurações;
20. Assim que terminar com sucesso, clique em **Close**.

D – Assinando a publicação Transactional

1. Ainda na primeira instância, no servidor **Publisher**, expanda a pasta **Local Publications** e observe, do lado esquerdo da tela, no **Object Explorer**, que a publicação de nome **PublicacaoTransacional_Ponto_A_Ponto** já faz parte da lista de publicações;
2. Clique com o botão direito do mouse sobre a publicação de nome **PublicacaoTransacional_Ponto_A_Ponto** e escolha a opção **New Subscriptions**. Em seguida, clique em **Next**;
3. Na tela que será exibida, selecione a publicação de nome **PublicacaoTransacional_Ponto_A_Ponto** e clique no botão **Next**;
4. Na próxima tela, aceite a primeira opção selecionada como default e clique em **Next**;
5. Na tela seguinte, clique no botão **Add Subscriber** e selecione a opção **Add SQL Server Subscriber**;
6. Surgirá a tela de conexão. No campo **Server name**, selecione o nome da segunda instância, faça a conexão com a autenticação do Windows e clique em **Connect**;
7. Note que, na tela anterior, ficou selecionado apenas o nome da segunda instância. Aceite o default e clique em **Next**;
8. Na próxima tela, no campo **Connection to Subscriber**, clique no botão com as reticências (...);
9. Na tela que será exibida, selecione a opção **Run under the SQL Server Agent service account** e clique no botão **OK**;
10. Clique em **Next**;
11. Note que no campo **Agent Schedule** está selecionada a opção **Run continuously**. Clique em **Next**;

12. Na próxima tela, note que, no campo **Initialize when**, está selecionada a opção **Simultaneously commit changes**. Em seguida, clique em **Next**;
13. Na tela seguinte, escolha a opção **Use a linked server that you have already defined**. Depois, clique em **Next**;
14. Na nova tela, no campo **Initialize when**, note que está selecionada a opção **Immediately**. Aceite o default e clique em **Next**;
15. Na próxima tela, deixe selecionada a opção **Create Subscription(s)** e clique em **Next**;
16. Na tela seguinte, observe o relatório de tudo o que será feito pelo SQL Server e clique em **Finish**;
17. Observe o SQL Server executando todas as configurações;
18. Assim que tudo terminar com sucesso, clique em **Close**.

E – Executando as verificações

1. Expanda a segunda instância e a pasta **Databases**;
2. Clique com o botão direito do mouse sobre a pasta **Databases** e escolha a opção **Refresh**;
3. Expanda o database **BancoTransact_Dois**;
4. Expanda a pasta **Tables** e note que a tabela **Material** já foi replicada para a segunda instância;
5. Clique com o botão direito do mouse sobre a tabela **Material** e escolha a opção **Open Table**;
6. Observe que cinco materiais foram replicados para esta tabela, cujos dados foram enviados do servidor **Editor (Publisher)**;

7. Clique novamente sobre o nome do servidor da primeira instância e, no menu de opções, clique em **File / Open** e selecione **File** novamente;
8. Abra a pasta **Capítulo_09** e dê um duplo-clique sobre o **script_10_Transact_Dois**;
9. Aparecerá a tela de conexão. No campo **Server name**, escolha o nome da primeira instância e, com a autenticação do Windows, clique no botão **Connect**;
10. O **script_10_Transact_Dois** vai inserir mais cinco registros na tabela **Material** da primeira instância. Pressione F5 para executá-lo;
11. Aguarde alguns instantes e, no **Object Explorer**, na segunda instância, clique com o botão direito do mouse sobre a tabela **Material**. Escolha a opção **Open Table** e, na barra de ferramentas, clique no ícone que representa um ponto de exclamação (!) para que a tabela seja lida novamente;
12. Note que os cinco registros novos já foram recebidos pelo servidor **Assinante**;
13. Ainda na segunda instância, na barra de menus, escolha as opções **File / Open** e, em seguida, **File** novamente. Aplique um duplo-clique no **script_11_Transact_Dois**;
14. Na tela que será exibida, no campo **Server name**, selecione o nome da segunda instância e conecte-se com a autenticação do Windows. Em seguida, clique no botão **Connect**;
15. O **script_11_Transact_Dois** insere mais cinco registros na tabela **Material** do servidor **Assinante**. Esses registros serão enviados para o servidor **Publisher** (como acontece no caso da publicação do tipo **Merge**). Pressione F5 para executar esse script;
16. Aguarde alguns instantes para que os jobs de replicação sejam executados novamente;

17. Na primeira instância, expanda a pasta **Databases**;
18. Expanda o database **BancoTransact_Dois** e a pasta **Tables**;
19. Clique com o botão direito do mouse sobre a tabela **Material** e escolha a opção **Open Table**. Note que os dados inseridos no servidor **Assinante (Subscriber)** foram enviados para o servidor **Editor (Publisher)**;
20. Nas duas instâncias, feche todas as queries que foram abertas e também todas as pastas que tenham sido abertas;
21. Para encerrar, feche (minimize) as duas instâncias.



10

Gerenciando um banco de dados

- ✓ Auditoria;
- ✓ Checklist das atividades de um DBA;
- ✓ Revisão da conectividade do ambiente;
- ✓ Monitoração do ambiente.

10.1. Introdução

O gerenciamento de banco de dados envolve um amplo conjunto de aspectos que devemos observar. Neste capítulo, abordaremos tópicos importantes na administração do banco de dados, envolvendo as atividades periódicas de manutenção e ajustes para melhorar a performance do banco (tuning). É importante entendermos que essa atividade de administração deve buscar ser proativa, no intuito de reduzir ao máximo os incidentes que possam ocorrer no dia a dia de um banco de dados.

10.2. Auditoria

Auditoria é uma das atividades que o administrador de banco de dados deve realizar. Ela pode ser agendada para ser executada de forma periódica ou pontual.

Podemos classificar a auditoria segundo os seguintes tipos:

- Auditoria de objetos;
- Auditoria de segurança;
- Auditoria de instância/servidor;
- Auditoria de dados.

A auditoria de banco de dados pode ter diversos focos de atuação, mas vamos tratar apenas dos mais usuais. Nos próximos subtópicos, abordaremos mais detalhadamente a auditoria de objetos e de segurança.

Para que a auditoria tenha mais chances de ser bem-sucedida, recomenda-se que ela possua um foco claro, determinado e, preferencialmente, bem limitado, pois pode despender recursos computacionais e profissionais e não chegar aos objetivos previstos.

10.2.1. Auditoria de objetos

A auditoria de objetos envolve controlar ou monitorar objetos criados, alterados ou eliminados. Neste tipo de auditoria, o foco é capturar as alterações realizadas nas estruturas do banco de dados, logo, não se procura auditar os dados em si, mas o que suporta e mantém esses dados. Por exemplo, quando desejamos saber se algum usuário está eliminando e recriando uma tabela, ou ainda se alterou algum procedimento.

Neste caso, o foco é controlar o objeto e não o que ele contém. Para auditar objetos, dispomos de duas formas de auditoria, a saber:

- Através de gatilhos (especificamente TRIGGERS DDL);
- Através do recurso de auditoria de banco de dados (AUDIT DATABASE).

10.2.1.1. Gatilhos

Gatilhos (TRIGGERS) de banco de dados podem ser divididos em gatilhos do tipo DML, InsteadOf ou DDL/Sistema. Os gatilhos DML estão ligados às ações de DML que ocorrem no banco de dados, os quais não atendem à perspectiva de auditoria de objetos. Os gatilhos DDL, por sua vez, são recursos que podem ser utilizados para auditar e monitorar alterações em schemas. Assim, todos os comandos DDL podem ser auditados através desse recurso.

Existem duas categorias de gatilhos DDL:

- **Transact DDL trigger:** Utiliza a linguagem Transact SQL para construção do gatilho;
- **CLR DDL Trigger:** Utiliza as linguagens da arquitetura .NET para construção dos gatilhos.

SQL 2014 - Módulo III

Veja a sintaxe de um gatilho DDL:

```
CREATE TRIGGER trigger_name
ON { ALL SERVER | DATABASE }
[ WITH <ddl_trigger_option> [ ,...n ] ]
{ FOR | AFTER } { event_type | event_group } [ ,...n ]
AS { sql_statement [ ; ] [ ,...n ] | EXTERNAL NAME < method
specifier > [ ; ] }
```



```
<ddl_trigger_option> ::==
[ ENCRYPTION ]
[ EXECUTE AS Clause ]
```

Agora, veja um exemplo de gatilho de auditoria usando Transact-SQL:

```
CREATE TABLE tab_ddl_log (nomeprograma varchar(100), datahora
datetime, usuario varchar(100), Evento nvarchar(100), comando
nvarchar(2000));
GO
```

```
IF EXISTS (SELECT * FROM sys.triggers
           WHERE parent_class = 0 AND name = 'monitora.Alter_DDL')
DROP TRIGGER monitora.Alter_DDL
ON DATABASE;
GO

CREATE TRIGGER monitora.Alter_DDL
ON DATABASE
FOR alter_table
AS
SET ANSI_PADDING ON
DECLARE @data XML
SET @data = EVENTDATA()
INSERT tab_ddl_log
    (nomeprograma, datahora, usuario, evento, comando)
VALUES
    (program_name(), GETDATE(),
    suser_sname(),
    @data.value('(/EVENT_INSTANCE/EventType)[1]',
    'nvarchar(100)'),
    @data.value('(/EVENT_INSTANCE/TSQLCommand)[1]',
    'nvarchar(2000)') );
```

Para validar esse gatilho, utilize este código:

```
create table teste (codigo int)
alter table teste add nome varchar(50)

select * from tab_ddl_log
```

O exemplo a seguir impede a criação de novas tabelas:

```
CREATE TRIGGER impede_criacao_tabela
ON DATABASE
FOR CREATE_TABLE
AS
    PRINT 'Comando Create table avaliado.'
    SELECT EVENTDATA().value
        ('(/EVENT_INSTANCE/TSQLCommand/CommandText)
[1]', 'nvarchar(max)')
    RAISERROR ('Não é possível a criação de tabelas neste banco de
dados.', 16, 1)
    ROLLBACK
```

Para validar esse gatilho, utilize o seguinte código:

```
create table teste2 (codigo int)
```

A imagem a seguir será retornada:

```
Comando Create table avaliado.

(1 row(s) affected)
Msg 50000, Level 16, State 1, Procedure impede_criacao_tabela, Line 8
Não é possível a criação de tabelas neste banco de dados.
Msg 3609, Level 16, State 2, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

Para capturar todos os eventos em uma tabela, utilize o seguinte código:

```
CREATE TRIGGER monitora.Alter_DDL
ON DATABASE
FOR alter_table ,create_table,drop_table
AS
SET ANSI_PADDING ON
DECLARE @data XML
```

SQL 2014 - Módulo III

```
SET @data = EVENTDATA()
INSERT tab_ddl_log
    (nomeprograma, datahora, usuario, evento, comando)
VALUES
    (program_name(), GETDATE(),
    suser_sname(),
    @data.value('(/EVENT_INSTANCE/EventType)[1]',
    'nvarchar(100)'),
    @data.value('(/EVENT_INSTANCE/TSQLCommand)[1]',
    'nvarchar(2000)') );
```

Para validar esse gatilho, use este código:

```
create table teste (codigo int)
alter table teste add nome varchar(50)

select * from tab_ddl_log
```

Veja, a seguir, como capturar alterações em procedimentos:

```
CREATE TABLE dbo.tab_eventos_ddl
(
    DataEvento      DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    TipoEvento     NVARCHAR(64),
    DDLEvento      NVARCHAR(MAX),
    XMLEvento      XML,
    NomeBD          NVARCHAR(255),
    NomeSchema      NVARCHAR(255),
    NomeObjeto      NVARCHAR(255),
    NomeServidor    VARCHAR(64),
    EnderecoIP      VARCHAR(32),
    NomePrograma    NVARCHAR(255),
    NomeLogin       NVARCHAR(255)
);
```

Veja, a seguir, como criar o gatilho DDL:

```
CREATE TRIGGER GATILHO_DDL_PROCEDURE
    ON DATABASE
--WITH ENCRYPTION

    FOR CREATE PROCEDURE, ALTER PROCEDURE, DROP PROCEDURE
AS
```

```
BEGIN
    SET NOCOUNT ON;
    DECLARE
        @EventData XML = EVENTDATA();

    DECLARE
        @ip VARCHAR(32) =
        (
            SELECT client_net_address
            FROM sys.dm_exec_connections
            WHERE session_id = @@SPID
        );

    INSERT dbo.tab_eventos_ddl
    (
        TipoEvento,
        DDLEvento,
        XMLEvento,
        NomeBD,
        NomeSchema,
        NomeObjeto,
        NomeServidor,
        EnderecoIP,
        NomePrograma,
        NomeLogin
    )
    SELECT
        @EventData.value('(/EVENT_INSTANCE/EventType)[1]',
        'NVARCHAR(100)'),
        @EventData.value('(/EVENT_INSTANCE/TSQLCommand)[1]',
        'NVARCHAR(MAX)'),
        @EventData,
        DB_NAME(),
        @EventData.value('(/EVENT_INSTANCE/SchemaName)[1]',
        'NVARCHAR(255)'),
        @EventData.value('(/EVENT_INSTANCE/ObjectName)[1]',
        'NVARCHAR(255)'),
        HOST_NAME(),
        @ip,
        PROGRAM_NAME(),
        SUSER_SNAME();
END
GO
```

Podemos, ainda, realizar a encriptação do gatilho (cláusula WITH ENCRYPTION), impedindo, assim, que ele possa ser visto mesmo que por usuários mais privilegiados. Neste caso, é necessário guardar o código-fonte em um local seguro, pois, em caso de necessidade de manutenção, apenas esse código poderá ser utilizado.

10.2.1.2. AUDIT DATABASE

Este recurso permite a auditoria em um banco de dados e pode ocorrer na instância e nos bancos de dados.

O seguinte comando permite criar uma auditoria que indique o diretório em que o arquivo de auditoria será gerado:

```
USE master  
GO  
CREATE SERVER AUDIT AUDIT_NAME TO file (filepath = 'FILE');
```

Nosso próximo passo deve ser criar uma especificação para a auditoria, na qual indicaremos o(s) comando(s) a ser(em) auditado(s):

```
USE DBNAME  
GO  
CREATE DATABASE AUDIT SPECIFICATION AUDIT_SPEC_NAME  
FOR SERVER AUDIT AUDIT_NAME  
    ADD (SELECT ON TABLE_NAME BY SCHEMA_NAME) WITH (STATE = ON);  
GO
```

10.2.2. Auditoria de segurança

Outro recurso que podemos usar em gatilhos DDL é o ALL SERVER, que permite que o gatilho ocorra em qualquer banco de dados no servidor. Para exemplificar, vamos auditar todas as conexões realizadas no banco de dados.

Veja a sintaxe para este tipo de gatilho:

```
CREATE TRIGGER trigger_name
ON ALL SERVER
[ WITH <logon_trigger_option> [ ,...n ] ]
{ FOR | AFTER } LOGON
AS { sql_statement [ ; ] [ ,...n ] | EXTERNAL NAME < method
specifier > [ ; ] }

<logon_trigger_option> ::==
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]
```

Para consultar os eventos que provocam o disparo dos gatilhos, utilize o seguinte código:

```
SELECT A.*
FROM sys.trigger_events AS A
JOIN sys.triggers AS B
ON A.object_id = B.object_id
WHERE B.parent_class = 0
```

Veja como desabilitar um gatilho:

```
DISABLE TRIGGER NOME_GATILHO ON DATABASE;
```

Agora, veja como habilitar um gatilho:

```
ENABLE TRIGGER NOME_GATILHO ON DATABASE;
```

E, por último, veja como eliminar um gatilho:

```
DROP TRIGGER NOME_GATILHO ON DATABASE;
```

10.3. Checklist de atividades de um DBA

Podemos definir a frequência das atividades de um administrador de banco de dados SQL Server como diárias, semanais, quinzenais e mensais. A seguir, listaremos o checklist básico correspondente a essas atividades.

10.3.1. Atividades diárias

É recomendável que as atividades sejam de caráter diário, pois deixar de executá-las por um ou mais dias pode implicar problemas para o ambiente ou para a recuperação deste em caso de falhas.

- **Revisão dos backups diários**

É importante revisar se todos os backups diários foram realizados. Caso tenha ocorrido alguma falha, é importante que eles sejam reexecutados o mais breve possível. Recomenda-se verificar a execução das tarefas. Em caso de falha, é possível confirmar se uma tarefa foi ou não executada.

O script a seguir pode auxiliar a verificar os backups executados no dia anterior:

```
SELECT
    A.[Server],
    A.last_db_backup_date,
    B.backup_start_date,
    B.expiration_date,
    B.backup_size,
    B.logical_device_name,
    B.physical_device_name,
    B.backupset_name,
    B.description
FROM
    (
        SELECT
            CONVERT(CHAR(100), SERVERPROPERTY('Servername')) AS Server,
            msdb.dbo.backupset.database_name,
            MAX(msdb.dbo.backupset.backup_finish_date) AS last_db_
        backup_date
        FROM      msdb.dbo.backupmediafamily
```

```
    INNER JOIN msdb.dbo.backupset ON msdb.dbo.
backupmediafamily.media_set_id = msdb.dbo.backupset.media_set_id
    WHERE msdb..backupset.type = 'D'
    GROUP BY
        msdb.dbo.backupset.database_name
) AS A

LEFT JOIN

(
SELECT
    CONVERT(CHAR(100), SERVERPROPERTY('Servername')) AS Server,
    msdb.dbo.backupset.database_name,
    msdb.dbo.backupset.backup_start_date,
    msdb.dbo.backupset.backup_finish_date,
    msdb.dbo.backupset.expiration_date,
    msdb.dbo.backupset.backup_size,
    msdb.dbo.backupmediafamily.logical_device_name,
    msdb.dbo.backupmediafamily.physical_device_name,
    msdb.dbo.backupset.name AS backupset_name,
    msdb.dbo.backupset.description
FROM msdb.dbo.backupmediafamily
    INNER JOIN msdb.dbo.backupset ON msdb.dbo.backupmediafamily.
media_set_id = msdb.dbo.backupset.media_set_id
    WHERE msdb..backupset.type = 'D'
) AS B
ON A.[server] = B.[server] AND A.[database_name] = B.[database_
name] AND A.[last_db_backup_date] = B.[backup_finish_date]
ORDER BY
    A.database_name
```

- **Processamento de tarefas noturnas**

É importante comprovar sempre se as atividades noturnas foram realizadas. O script a seguir poderá ajudar nessa atividade. Lembre-se de definir neste script o nome das tarefas que precisa verificar:

```
Select
*
From
    msdb..sysjobhistory as sysjobhistory
    Join msdb..sysjobs as sysjobs on sysjobhistory.job_
id=sysjobhistory.job_id
Where
    Name='nome da tarefa'
```

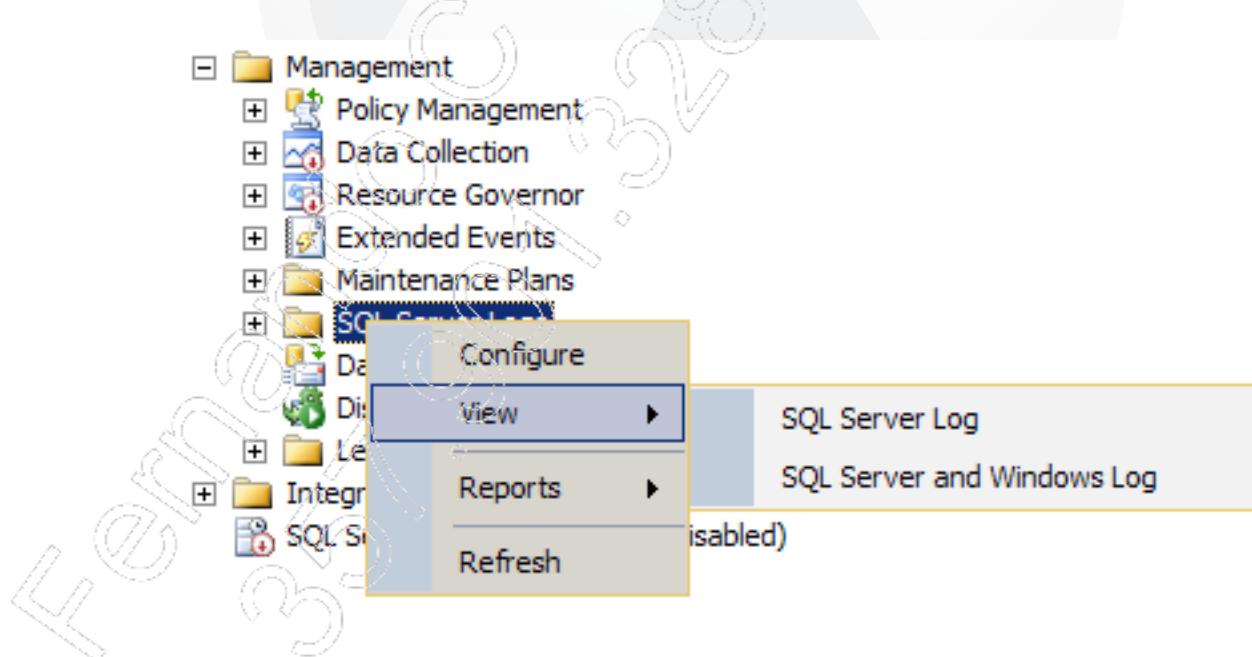
```
Order By  
    Run_Date Desc,  
    run_time Desc
```

- **Verificação do SQL Server errorlog**

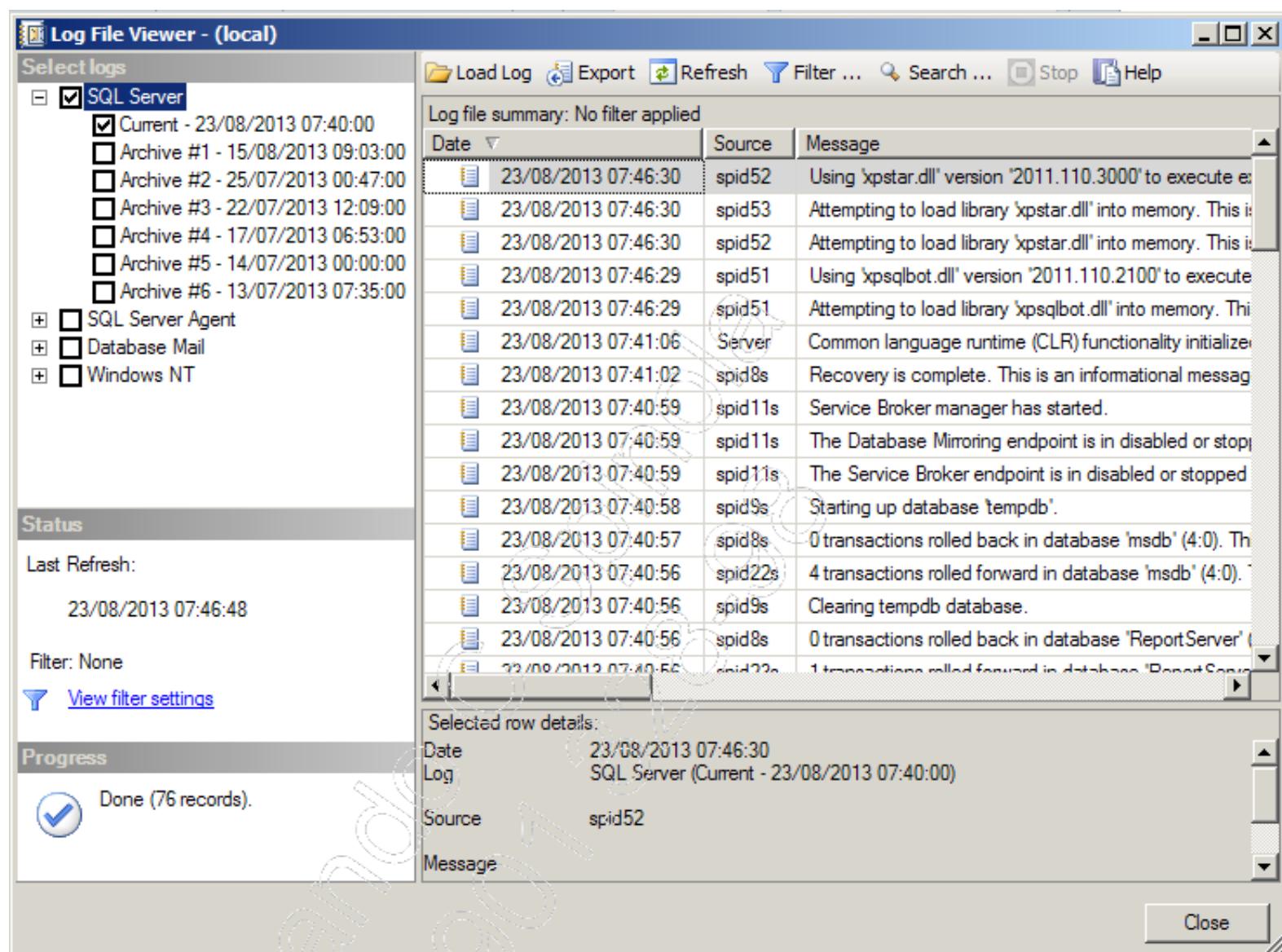
O log de erros do SQL Server registra as atividades e falhas que ocorrem. É importante analisarmos se ocorreram erros ou falhas graves. Estes arquivos são gravados no seguinte diretório: **Program Files\Microsoft SQL Server\MSSQL.n\MSSQL\LOG\ERRORLOG and ERRORLOG.n**.

Não podemos abrir o arquivo atual. Para forçar a troca do arquivo atual para a criação de um novo, devemos usar a procedure **sp_cycle_errorlog**. Com isso, o log atual passa a ser o anterior e um novo log é aberto no seu lugar (log corrente).

1. No SSMS, clique com o botão direito do mouse sobre a opção **SQL Server Log**, abra a opção **View** e escolha **SQL Server Log**:



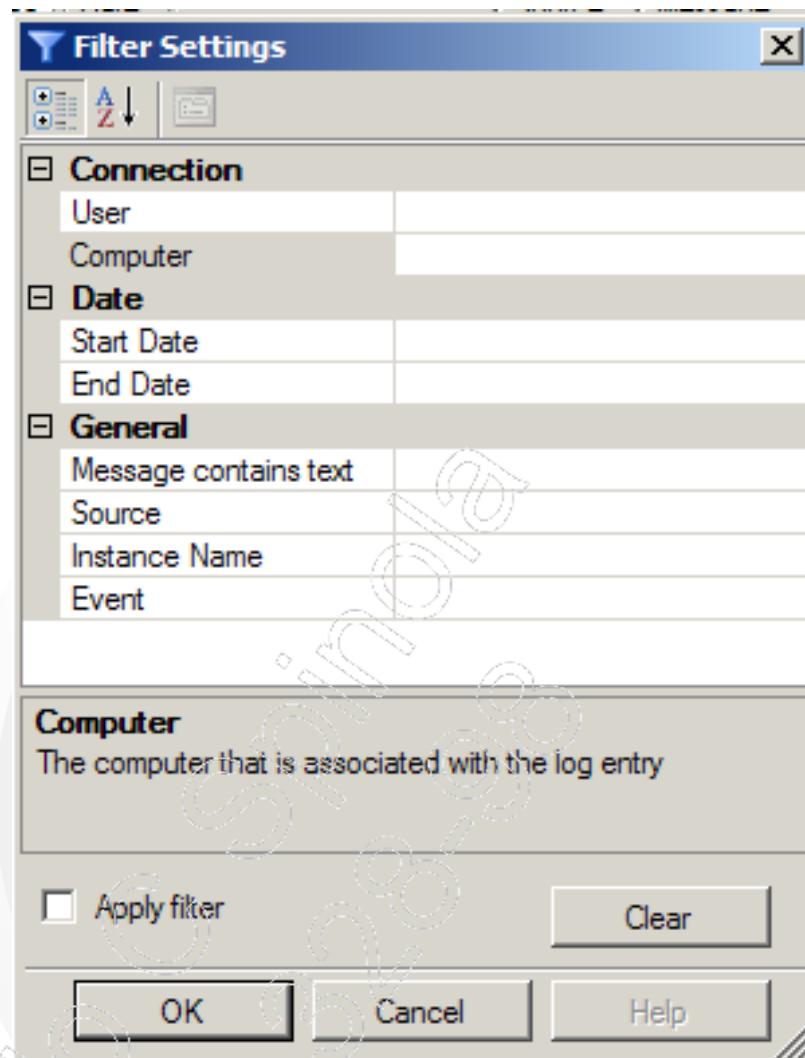
2. A tela a seguir será exibida. Nela, selecione a opção **Current** para exibir o conteúdo do errorlog atual:



Quando clicamos na opção **SQL Server** dessa janela, podemos visualizar o errorlog do SQL Server.

Para verificar o errorlog do SQL Server Agent, do Database Mail e do Event Viewer Windows, devemos selecionar, respectivamente, estas opções: **SQL Server Agent**, **Database Mail** e **Windows NT**.

3. Selecione o botão **Filter** para realizar um filtro de pesquisa. Dessa forma, a tela a seguir será exibida:



4. Selecione a opção **Start Date** e indique a data do dia de ontem. Na opção **End Date**, escreva o dia de hoje e pressione o botão **OK**. Apenas as ocorrências dos dias selecionados serão apresentadas. Também é possível realizar a pesquisa pelos seguintes critérios além das datas de eventos:

- **Message contains text:** Texto que deseja ser pesquisado no errorlog;
- **Source:** Origem do erro;
- **Instance Name:** Nome da instância;
- **Event:** Código do evento a ser pesquisado (Event ID).

O script a seguir permite identificar erros críticos de log e enviá-los por e-mail:

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[dba_all_errorlog_details] (
    [id] [INT] IDENTITY(1,1) NOT NULL,
    [date] [DATETIME] NULL,
    [processinfo] [SYSNAME] NOT NULL,
    [text] [SYSNAME] NOT NULL
) ON [PRIMARY]

GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROC [dbo].[prc_dba_critical_error]
AS
BEGIN
    --TRUNCATE TABLE DBA_test.DBO.dba_all_errorlog_details

    INSERT INTO DBA_test.DBO.dba_all_errorlog_details
    EXEC DBA_test.DBO.SP_READERRORLOG 0,1,
    'Setting database option OFFLINE'

    INSERT INTO DBA_test.DBO.dba_all_errorlog_details
    EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'killed by'

    INSERT INTO DBA_test.DBO.dba_all_errorlog_details
    EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'alter database'

    INSERT INTO DBA_test.DBO.dba_all_errorlog_details
    EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'setting database option
    recovery'

    INSERT INTO DBA_test.DBO.dba_all_errorlog_details
    EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'account is currently
    locked out'
```

SQL 2014 - Módulo III

```
INSERT INTO DBA_test.DBO.dba_all_errorlog_details
EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'sql server is terminating
due to'

INSERT INTO DBA_test.DBO.dba_all_errorlog_details
EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'deadlock'

INSERT INTO DBA_test.DBO.dba_all_errorlog_details
EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'the log is out of space'

INSERT INTO DBA_test.DBO.dba_all_errorlog_details
EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'error: 9002'

INSERT INTO DBA_test.DBO.dba_all_errorlog_details
EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'severity: 13'

INSERT INTO DBA_test.DBO.dba_all_errorlog_details
EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'severity: 17'

INSERT INTO DBA_test.DBO.dba_all_errorlog_details
EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'severity: 18'

INSERT INTO DBA_test.DBO.dba_all_errorlog_details
EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'severity: 19'

INSERT INTO DBA_test.DBO.dba_all_errorlog_details
EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'severity: 20'

INSERT INTO DBA_test.DBO.dba_all_errorlog_details
EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'severity: 21'

INSERT INTO DBA_test.DBO.dba_all_errorlog_details
EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'severity: 22'

INSERT INTO DBA_test.DBO.dba_all_errorlog_details
EXEC DBA_test.DBO.SP_READERRORLOG 0,1,'severity: 23'

--SELECT * FROM DBA_test.DBO.dba_all_errorlog_details
END

-- Exec [usp_dba_critical_error_DBmail]

USE [DBA_test]
GO
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROC [dbo].[prc_dba_critical_error_DBmail]
AS
BEGIN
EXEC [DBA_test].[dbo].[prc_dba_critical_error]

IF EXISTS(SELECT 1 FROM DBO.dba_all_errorlog_details )
BEGIN
DECLARE @processinfo VARCHAR(500)
DECLARE @text VARCHAR(5000)
DECLARE @date VARCHAR(200)
DECLARE @dateadd VARCHAR(200)
DECLARE @maxid int
DECLARE @minid int
SET @dateadd = REPLACE(CONVERT(CHAR(8),DATEADD(HH,-
2,GETDATE()),108),':','');
select @minid=MIN(id) from DBA_test.DBO.dba_all_errorlog_details
select @maxid=MAX(id) from DBA_test.DBO.dba_all_errorlog_details
WHILE (@minid<=@maxid)
BEGIN
SELECT @date=[date],@processinfo=[processinfo],@text=[text] FROM
DBA_test.DBO.dba_all_errorlog_details
WHERE id=@minid and REPLACE(CONVERT(CHAR(8),DATEADD(hh,-
2,date),108),':','') < @dateadd
set @minid=@minid+1
END
DECLARE @body1 VARCHAR(2000)
SET @body1= 'server :Mensagens de erros críticas '+ CHAR(13)
+CHAR(13)
SET @body1= @body1 +
'DATE: '+@date+CHAR(9)+ 
'PROCESSINFO: '+@processinfo+ CHAR(13)+ 
'TEXT:' +@text+ CHAR(13)+CHAR(13)
EXEC MSDB.DBO.SP_SEND_DBMAIL @recipients='Informe o email
desejado',
@subject = 'server : Mensagens de erros críticas',
@body = @body1,
@body_format = 'text' ,@profile_name='NOME PROFILE BACKUP';
END
END
```

- **Checagem da alta disponibilidade do ambiente**

É importante a análise diária da alta disponibilidade do ambiente, pois, em caso de falha, é com essa análise que devemos contar para o reestabelecimento do ambiente. Examine o estado do log shipping, do cluster, da replicação e demais recursos para a alta disponibilidade.

- **Análise de segurança do banco de dados**

Verifique a segurança do banco de dados. Os scripts a seguir auxiliam a buscar informações importantes relativas à segurança do banco de dados.

```
DECLARE @AuditLevel int
EXEC master.dbo.xp_instance_regread N'HKEY_LOCAL_MACHINE',
    N'Software\Microsoft\MSSQLServer\MSSQLServer',
    N'AuditLevel', @AuditLevel OUTPUT
SELECT CASE WHEN @AuditLevel = 0 THEN 'None'
    WHEN @AuditLevel = 1 THEN 'Successful logins only'
    WHEN @AuditLevel = 2 THEN 'Failed logins only'
    WHEN @AuditLevel = 3 THEN 'Both failed and successful logins'
    END AS [AuditLevel]
```

Esse script mostra o nível de auditoria que está sendo registrado nos errorlogs (um para todos os logins bem-sucedidos, dois para os logins que apresentaram falhas e três para ambos).

Configure o número de errorlogs para um número superior a 20, pois, como os logs sofrem um processo de reaproveitamento cíclico, um número baixo pode gerar um período muito pequeno de logs.

O script a seguir muda o regedit para que sejam gerados até 30 logs; cuidado apenas para definir a instância correta (sublinhada):

```
EXEC master.dbo.xp_instance_regwrite N'HKEY_LOCAL_MACHINE',
    N'Software\Microsoft\MSSQLServer\MSSQLServer',
    N'NumErrorLogs', REG_DWORD, 30
```

Veja como verificar os logins que não foram executados (falhas):

```
EXEC master.dbo.xp_readerrorlog 0, 1, 'login failed', null, NULL,  
NULL, N'desc'
```

E, agora, veja como verificar os usuários que foram removidos da conta de administradores:

```
SELECT r.name as SrvRole, u.name as LoginName  
FROM sys.server_role_members m JOIN  
    sys.server_principals r ON m.role_principal_id = r.principal_id  
JOIN  
    sys.server_principals u ON m.member_principal_id = u.principal_id  
WHERE u.name = 'BUILTIN\Administrators'
```

Para mapear logins que foram criados no **schema dbo**, utilize este código:

```
EXEC master.sys.sp_MSforeachdb '  
PRINT ''?''  
EXEC [?].dbo.sp_helpuser ''dbo'''
```

Ainda podemos verificar as políticas de senha e quando elas expiram:

```
SELECT name FROM sys.sql_logins  
WHERE is_policy_checked=0 OR is_expiration_checked = 0
```

Já o código adiante permite verificar as configurações definidas para a instância, tais como:

- Database Mail;
- Allow updates;
- Xp_cmdshell.

```
SELECT name, value_in_use FROM sys.configurations  
WHERE configuration_id IN (16391, 102, 400, 1562, 16386, 16385,  
16390, 16393)
```

SQL 2014 - Módulo III

E este código permite verificar usuários que podem se conectar com as permissões de GUEST:

```
SET NOCOUNT ON
CREATE TABLE #guest_perms
( db SYSNAME, class_desc SYSNAME,
  permission_name SYSNAME, ObjectName SYSNAME NULL)
EXEC master.sys.sp_MSforeachdb
`INSERT INTO #guest_perms
SELECT ''?'' as DBName, p.class_desc, p.permission_name,
  OBJECT_NAME (major_id, DB_ID('?'')) as ObjectName
  FROM [?].sys.database_permissions p JOIN [?].sys.database_
principals l
    ON p.grantee_principal_id= l.principal_id
 WHERE l.name = 'guest' AND p.[state] = 'G'

SELECT db AS DatabaseName, class_desc, permission_name,
  CASE WHEN class_desc = 'DATABASE' THEN db ELSE ObjectName END as
ObjectName,
  CASE WHEN DB_ID(db) IN (1, 2, 4) AND permission_name = 'CONNECT'
THEN 'Default'
  ELSE 'Potential Problem!' END as CheckStatus
FROM #guest_perms
DROP TABLE #guest_perms
```

Para localizar os usuários que não estão associados a um login (ORPHAN USERS), utilizamos este código:

```
EXEC master.sys.sp_msforeachdb `'
print '?'
EXEC [?].dbo.sp_change_users_login ''report'''
```

- **Verificação de área disponível em disco**

É importante verificar o espaço em disco para os bancos de dados e para os logs do SQL server. Veja a seguir um script que faz essa verificação:

```
EXECUTE SP_CONFIGURE 'show advanced options', 1
RECONFIGURE WITH OVERRIDE
GO
EXECUTE SP_CONFIGURE 'xp_cmdshell', '1'
RECONFIGURE WITH OVERRIDE
GO
```

```
EXECUTE SP_CONFIGURE 'show advanced options', 0
RECONFIGURE WITH OVERRIDE
GO
```

```
declare @svrName varchar(255)
declare @sql varchar(400)
set @svrName = @@SERVERNAME
set @sql = 'powershell.exe -c "Get-WmiObject -ComputerName ' +
QUOTENAME(@svrName,'''') + ' -Class Win32_Volume
-Filter ''DriveType = 3'' | select name,capacity,freespace |
foreach{$_.name+'|'+$_ .capacity/1048576+'%' +$_ .
freespace/1048576+'*' }"'
CREATE TABLE #output
(line varchar(255))
insert #output
EXEC xp_cmdshell @sql
--script to retrieve the values in MB from PS Script output
select rtrim(ltrim(SUBSTRING(line,1,CHARINDEX('|',line) -1))) as
drivename
    ,round(cast(rtrim(ltrim(SUBSTRING(line,CHARINDEX('|',li
ne)+1,
        (CHARINDEX('%',line) -1)-CHARINDEX('|',line)) )) as
Float),0) as 'capacity(MB)'
    ,round(cast(rtrim(ltrim(SUBSTRING(line,CHARINDEX('%',li
ne)+1,
        (CHARINDEX('*',line) -1)-CHARINDEX('%',line)) )) as
Float),0) as 'freespace(MB)'
from #output
where line like '[A-Z] [:]%'
order by drivename
select rtrim(ltrim(SUBSTRING(line,1,CHARINDEX('|',line) -1))) as
drivename
    ,round(cast(rtrim(ltrim(SUBSTRING(line,CHARINDEX('|',li
ne)+1,
        (CHARINDEX('%',line) -1)-CHARINDEX('|',line)) )) as
Float)/1024,0) as 'capacity(GB)'
    ,round(cast(rtrim(ltrim(SUBSTRING(line,CHARINDEX('%',li
ne)+1,
        (CHARINDEX('*',line) -1)-CHARINDEX('%',line)) )) as Float)
/1024 ,0)as 'freespace(GB)'
from #output
where line like '[A-Z] [:]%'
order by drivename
drop table #output
```

- **Verificação do service broker**

Muitas aplicações utilizam o serviço broker do SQL Server, por isso, é importante que ele esteja operacional. Para habilitarmos o service broker, utilizamos a seguinte sintaxe:

```
ALTER DATABASE nome_banco_dados SET ENABLE_BROKER
```

- **Compactação do arquivo de log**

A compactação dos arquivos de log é uma atividade necessária quando o transaction log sofreu um aumento de tamanho significativo, podendo, em determinadas circunstâncias, ter ficado maior que os próprios arquivos de dados. Essa operação deve ser realizada após um backup de log. Veja a seguir o código necessário para realização desse procedimento:

```
select name,recovery_model_desc from sys.databases  
USE [master]  
GO  
ALTER DATABASE DBNAME SET RECOVERY SIMPLE WITH NO_WAIT  
GO  
  
DBCC Shrinkfile('NOMEARQUIVOLOG',1000)  
  
ALTER DATABASE DBNAME SET RECOVERY FULL WITH NO_WAIT
```

O valor informado no comando **Shrinkfile** é o tamanho desejado em MB.

10.3.2. Atividades semanais

Veja as atividades semanais que são recomendadas para que o ambiente de banco de dados esteja íntegro:

- Verificação de correções (Updates e Cumulative Updates) disponíveis para aplicação;
- Fragmentação de tabelas e índices;
- Compactação de arquivos de dados e de log;
- Análise de índices necessários e desnecessários no ambiente.

10.3.3. Atividades mensais

São recomendadas as seguintes atividades mensais:

- Verificação física de banco de dados (DBCC);
- Verificação dos backups realizados durante o mês;
- Rever o capacity plan para determinar a necessidade de mais recursos computacionais para o ambiente (servidor, memória, processador e armazenamento).

10.4. Revisão da conectividade do ambiente

É importante para o administrador sempre realizar a revisão da conectividade do ambiente no que diz respeito a logins, credenciais e linked servers, que são o ponto de conectividade entre diferentes instâncias/servidores no SQL Server.

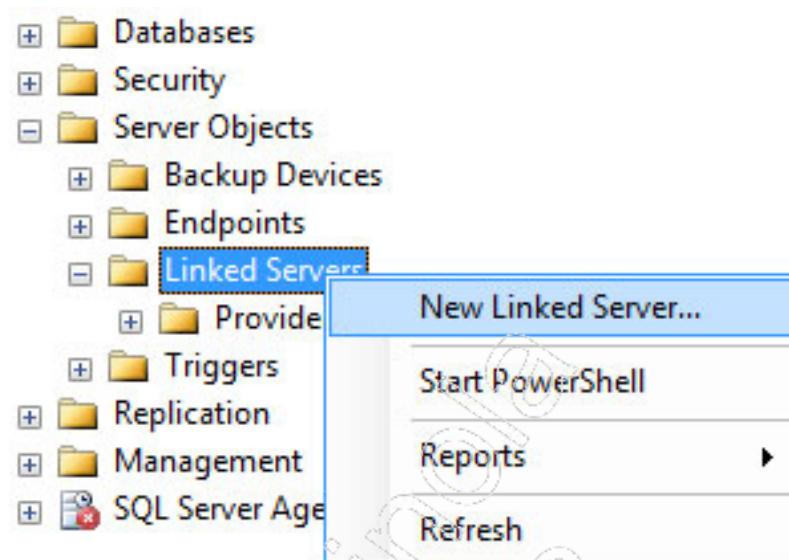
A revisão de usuários e logins faz parte do checklist diário/semanal do DBA. A conectividade entre instâncias é também um item a ser visto, já que uma falha nela pode implicar problemas junto às aplicações e, consequentemente, no banco de dados. Veja como criar um Linked Server por meio do código:

```
USE [master]
GO
EXEC master.dbo.sp_addlinkedserver
    @server = N'SRVR002\ACCTG',
    @srvproduct=N' SQL Server' ;
GO

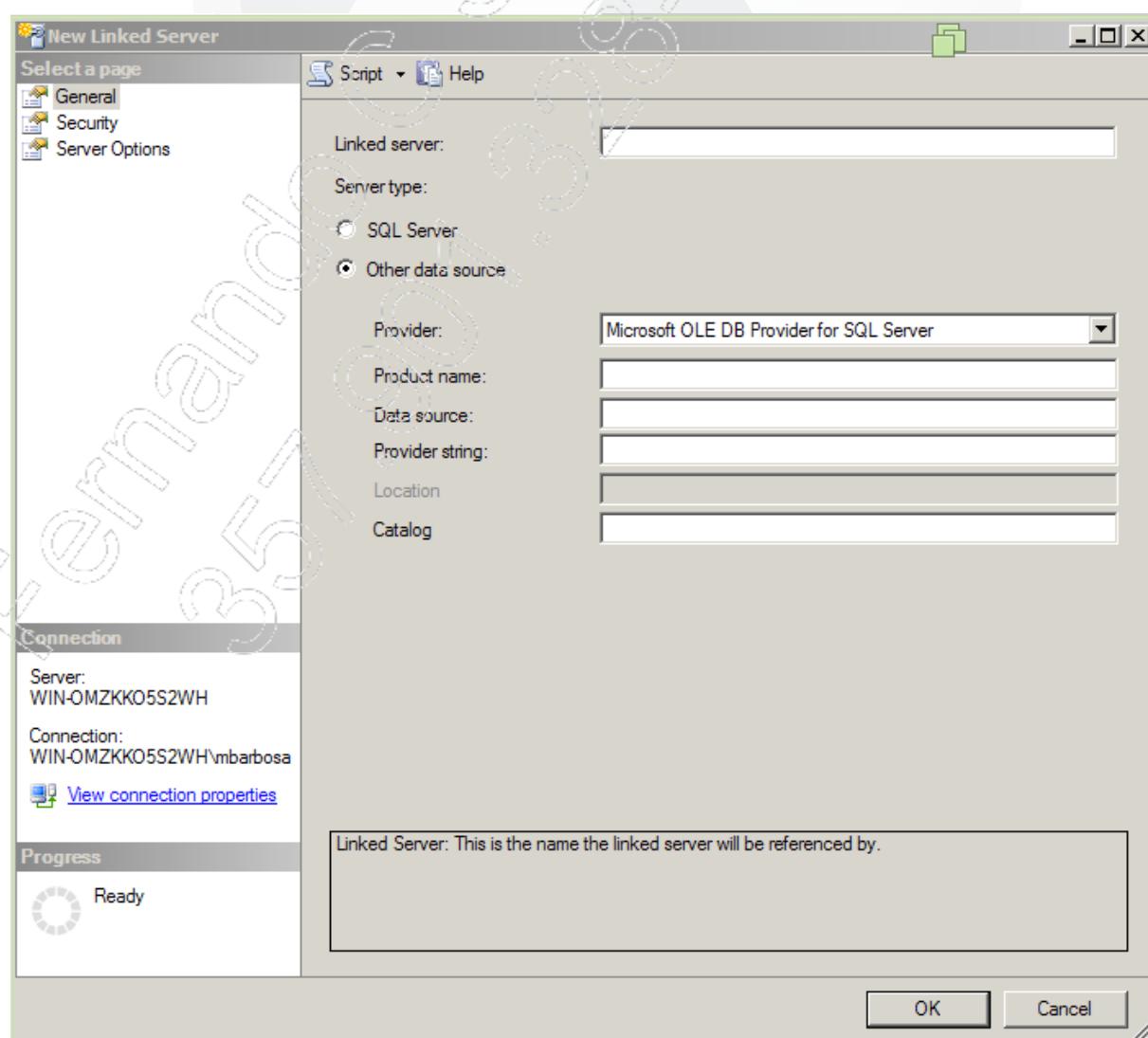
EXEC master.dbo.sp_addlinkedsrvlogin
    @rmtsrvname = N'SRVR002\ACCTG',
    @locallogin = NULL ,
    @useself = N'True' ;
GO
```

É necessário criar o Linked Server e o login ligado a ele. Veja como fazer isso através do SSMS:

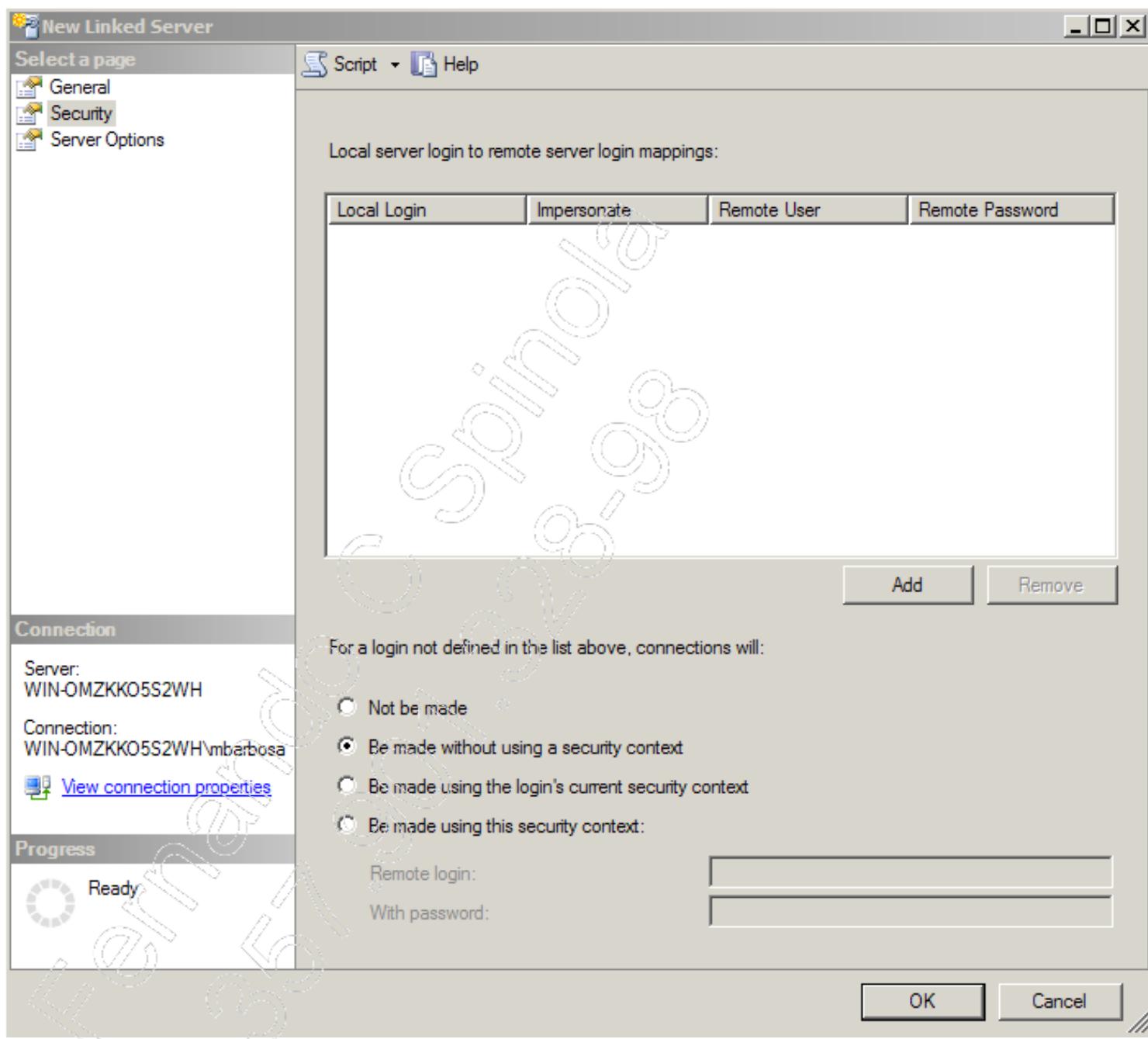
1. No SSMS, selecione a opção **Linked Server** e, em seguida, a opção **New Linked Server...**:



A seguinte tela será exibida:



2. Informe o nome do Linked Server, o tipo de conexão que irá realizar (SQL Server ou outro tipo de conexão) e o tipo de provedor de acesso que irá utilizar (Provider). Na pasta **Security**, selecione a opção **Be made without using a security context** e clique no botão **OK**:



10.5. Monitoração do ambiente

Monitoração do ambiente é uma atividade que o administrador precisa realizar para gerir os problemas que possam acontecer, mesmo que eles ainda não tenham ocorrido de fato.

Essa atividade pode ser feita com ferramentas de monitoração ou através da execução de Jobs para coleta de informações que são armazenadas em um banco de dados destinado a essa tarefa. A monitoração do ambiente ainda pode ser feita através do envio de e-mails quando for detectada alguma anormalidade.

Listaremos alguns dos pontos fundamentais de monitoração, lembrando que se trata apenas de uma ideia básica, que precisa e pode ser expandida.

Antes de tudo, para uma monitoração funcionar, precisamos entender que é necessário focar cada item a ser monitorado. Não podemos monitorar todo o ambiente com um único script de monitoração, é preciso dividir a monitoração em pequenas partes bastante objetivas e claras. Outro ponto importante é que toda monitoração deve originar um número e este deverá ser tratado usando-se alguns limites (thresholds) definidos pelo próprio administrador. Conforme o ambiente é possível estabelecer limites maiores (maior tolerância) ou menores (monitoração mais agressiva). Os limites devem ser tratados da seguinte forma:

Normal	0 a 75%
Aviso (Warning)	75,01 a 85%
Cuidado (Attention)	85,01 a 95%
Problema (Problem)	95 a 100%

Esses números podem mudar conforme o item monitorado e o grau de agressividade na monitoração do ambiente, geralmente ligado à criticidade desse ambiente (quanto mais crítico, mais agressivo).

Cada ponto de monitoração deve ser ligado a um script contendo uma consulta capaz de retornar um valor que deverá ser tratado conforme o limite em que se encaixa, para que, assim, seja feito o tratamento adequado ao problema.

Um ponto importante é o tempo em que esse trabalho de monitoração será feito (a cada quanto tempo será reexecutado). Cada ponto de monitoração tem o seu tempo: podemos estabelecer que as monitorações sejam executadas a cada minuto, a cada 5 minutos, 15 minutos, 30 minutos, 60 minutos, 120 minutos e 1440 minutos.

Outro ponto relevante é o destinatário da monitoração, ou seja, quem irá receber os avisos. Existe algum setor da empresa que realiza esta atividade? Em caso afirmativo, o destino da monitoração deve ser esse setor, que poderá analisar o problema e acionar quem tiver escalado para isso, caso ocorra um aviso fora do horário de trabalho. Caso o destino do aviso seja o administrador, de nada adianta avisá-lo fora do horário de trabalho, pois ele pode não ver o aviso a tempo e uma monitoração pode passar de normal a problemática em pouco tempo.

Veja, então, alguns dos itens importantes para monitoração:

- Espaço em disco;
- Tamanho do Transaction log;
- Realização de backups;
- Espaço nos arquivos de dados (se não estiverem com autoextensão);
- Alocação de memória;
- Monitoração de falhas informadas no errorlog;
- Shutdown de banco de dados;
- Parada de serviços SQL;
- Latch Waits (por segundo);
- Erros (por segundo).

A seguir, veja algumas consultas usadas em monitoração:

```
SELECT
    [Session ID] = s.session_id,
    [User Process] = CONVERT(CHAR(1), s.is_user_process),
    [Login] = s.login_name,
    [Database] = ISNULL(db_name(p.dbid), N''),
    [Task State] = ISNULL(t.task_state, N''),
    [Command] = ISNULL(r.command, N''),
    [Application] = ISNULL(s.program_name, N''),
    [Wait Time (ms)] = ISNULL(w.wait_duration_ms, 0),
    [Wait Type] = ISNULL(w.wait_type, N''),
    [Wait Resource] = ISNULL(w.resource_description, N''),
    [Blocked By] = ISNULL(CONVERT (varchar, w.blocking_session_id),
    ''),
    [Head Blocker] =
CASE
    WHEN r2.session_id IS NOT NULL AND (r.blocking_session_id = 0
    OR r.session_id IS NULL) THEN '1'
    --but is blocked by another party
    ELSE ''
END,
    [Total CPU (ms)] = s.cpu_time,
    [Total Physical I/O (MB)] = (s.reads + s.writes) * 8 / 1024,
    [Memory Use (KB)] = s.memory_usage * 8192 / 1024,
    [Open Transactions] = ISNULL(r.open_transaction_count,0),
    [Login Time] = s.login_time,
    [Last Request Start Time] = s.last_request_start_time,
    [Host Name] = ISNULL(s.host_name, N''),
    [Net Address] = ISNULL(c.client_net_address, N''),
    [Execution Context ID] = ISNULL(t.exec_context_id, 0),
    [Request ID] = ISNULL(r.request_id, 0),
    [Workload Group] = ISNULL(g.name, N'')
FROM sys.dm_exec_sessions s LEFT OUTER JOIN sys.dm_exec_
connections c ON (s.session_id = c.session_id)
LEFT OUTER JOIN sys.dm_exec_requests r ON (s.session_id =
r.session_id)
LEFT OUTER JOIN sys.dm_os_tasks t ON (r.session_id = t.session_id
AND r.request_id = t.request_id)
LEFT OUTER JOIN
(

```

SQL 2014 - Módulo III

```
SELECT *, ROW_NUMBER() OVER (PARTITION BY waiting_task_address
ORDER BY wait_duration_ms DESC) AS row_num
FROM sys.dm_os_waiting_tasks
) w ON (t.task_address = w.waiting_task_address) AND w.row_num =
1
LEFT OUTER JOIN sys.dm_exec_requests r2 ON (s.session_id =
r2.blocking_session_id)
LEFT OUTER JOIN sys.dm_resource_governor_workload_groups g ON
(g.group_id = s.group_id)
LEFT OUTER JOIN sys.sysprocesses p ON (s.session_id = p.spid)
ORDER BY s.session_id;
```

Por meio do código a seguir, o monitor de bloqueios verifica quais sessões estão sendo bloqueadas por outras sessões:

```
Select t1.spid ,t1.status, loginame=rtrim(t1.loginame),
hostname=LEFT(rtrim(t1.hostname),20), program_name=rtrim(t1.
program_name), t1.blocked,t1.dbid,
        dbname = rtrim(case when t1.dbid = 0 then null when
t1.dbid <> 0 then db_name(t1.dbid) end)),rtrim(t1.nt_username)nt_
username,
        rtrim(t1.cmd)cmd,
        datediff(minute,t1.last_batch,GETDATE()) waittime_dk,
        substring(sql.text, stmt_start/2,CASE WHEN stmt_end<1
THEN 8000 ELSE (stmt_end-stmt_start)/2 END) AS RunningSqlText,
        sql.text as FullSqlText,
        t1.cpu, substring( convert(varchar,t1.last_batch,111)
,6 ,5 ) +
        + substring( convert(varchar,t1.last_batch,113) ,13 ,8
)
        as 'last_batch_time',
        t1.waittime waittime,
        t1.lastwaittype
From master.dbo.sysprocesses (NOLOCK) t1
cross apply sys.dm_exec_sql_text(t1.sql_handle) AS sql
Where t1.blocked <> 0 OR t1.spid in (Select t2.blocked From
master.dbo.sysprocesses (NOLOCK) t2
Order By t1.dbid DESC,t1.spid
```

Para que o monitor de bloqueios verifique as sessões que bloqueiam as outras sessões, utilize este código:

```
SELECT    db_name(DTL.[resource_database_id]) AS [Database] ,
          DTL.[resource_type] AS [Resource Type] ,
          CASE WHEN DTL.[resource_type] IN ( 'DATABASE', 'FILE' ,
'METADATA' )
                THEN DTL.[resource_type]
                WHEN DTL.[resource_type] = 'OBJECT'
                THEN OBJECT_NAME(DTL.resource_associated_entity_id)
                WHEN DTL.[resource_type] IN ( 'KEY', 'PAGE', 'RID' )
                THEN ( SELECT OBJECT_NAME([object_id])
                      FROM sys.partitions
                      WHERE sys.partitions.[hobt_id] =
                            DTL.[resource_associated_entity_id]
                )
                ELSE 'Unidentified'
        END AS [Parent Object] ,
          DTL.[request_mode] AS [Lock Type] ,
          DTL.[request_status] AS [Request Status] ,
          DOWT.[wait_duration_ms] AS [Wait Duration (ms)] ,
          DOWT.[wait_type] AS [Wait Type] ,
          DOWT.[session_id] AS [Blocked Session ID] ,
          DES_Blocked.[login_name] AS [Blocked Login] ,
          SUBSTRING(DEST_Blocked.text, (DER.statement_start_offset
/ 2) + 1,
          ( CASE WHEN DER.statement_end_offset = -1
                  THEN DATALENGTH(DEST_Blocked.text)
                  ELSE DER.statement_end_offset
            END - DER.statement_start_offset ) / 2)
          AS [Blocked
Command] ,
          DOWT.[blocking_session_id] AS [Blocking Session ID] ,
          DES_Blocking.[login_name] AS [Blocking Login] ,
          DEST_Blocking.[text] AS [Blocking Command] ,
          DOWT.resource_description AS [Blocking Resource Detail]
FROM      sys.dm_tran_locks DTL
          INNER JOIN sys.dm_os_waiting_tasks DOWT
          ON DTL.lock_owner_address = DOWT.resource_
address
```

SQL 2014 - Módulo III

```
    INNER JOIN sys.[dm_exec_requests] DER
              ON DOWT.[session_id] = DER.[session_id]
    INNER JOIN sys.dm_exec_sessions DES_Blocked
              ON DOWT.[session_id] = DES_Blocked.[session_
id]
    INNER JOIN sys.dm_exec_sessions DES_Blocking
              ON DOWT.[blocking_session_id] = DES_Blocking.
[session_id]
    INNER JOIN sys.dm_exec_connections DEC
              ON DOWT.[blocking_session_id] = DEC.[most_-
recent_session_id]
    CROSS APPLY sys.dm_exec_sql_text(DEC.[most_recent_sql_-
handle])
AS DEST_-
Blocking
    CROSS APPLY sys.dm_exec_sql_text(DER.sql_handle) AS DEST_-
Blocked
```

Para monitorar o espaço em disco, utilize o seguinte código:

```
exec sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
exec sp_configure 'Ole Automation Procedures', 1;
GO
RECONFIGURE;
GO
DROP PROCEDURE [dbo].[PMON_DriveSpaceCheck]
CREATE PROCEDURE [dbo].[PMON_DriveSpaceCheck]
AS
BEGIN
DECLARE @chr INT ,
        @fso INT,
        @drive CHAR(1),
        @odrive INT,
        @TotalSize VARCHAR(20),
        @MB NUMERIC ,
        @FreeSpace INT,
        @free INT,
        @RowId_1 INT,
        @LoopStatus_1 SMALLINT,
        @TotalSpace VARCHAR(10),
        @Percentage VARCHAR(3)
```

```
--TABELA PARA ARMAZENAR INFORMAÇÕES DOS DISCOS

CREATE TABLE #drives
(
    id INT IDENTITY(1,1) PRIMARY KEY,
    drive CHAR(1),
    FreeSpaceMB INT ,
    TotalSizeMB INT NULL,
    percentage INT
)

--INSERINDO DADOS DA xp_fixeddrives NA TABELA #SpaceSize

INSERT #drives(drive,FreeSpaceMB) EXEC master.dbo.xp_fixeddrives

--Criando uma instância para o objeto OLE

EXEC @hr=sp_OACreate 'Scripting.FileSystemObject',@fso OUT

SET @MB = 1048576
SET @RowId_1 = 1
SET @LoopStatus_1 = 1

--Obtendo o espaço em disco

WHILE (@LoopStatus_1 <> 0) BEGIN

SELECT
@drive=drive,
```

SQL 2014 - Módulo III

```
@FreeSpace=FreeSpaceMB
FROM
#drives
WHERE
( ID = @RowId_1 )

IF ( @@ROWCOUNT = 0 )
BEGIN
SET @LoopStatus_1 = 0
END
ELSE
BEGIN
EXEC @hr = sp_OAMethod @fso,'GetDrive', @odrive OUT, @drive
EXEC @hr =sp_OAGetProperty @odrive,'TotalSize', @TotalSize OUT
UPDATE #drives SET TotalSizeMB=@TotalSize/@MB
WHERE
drive=@drive
UPDATE #drives SET Percentage=(@FreeSpace/
(TotalSizeMB*1.0))*100.0
WHERE drive=@drive
END
SET @RowId_1 = @RowId_1 + 1
END

SELECT * FROM #drives

DROP TABLE #drives
END

exec [dbo].[PMON_DriveSpaceCheck]
```

Para obter as informações de utilização dos arquivos de dados, utilize este código:

```
Use nome_database;
set nocount on
create table #Data(
    FileID int NOT NULL,
    [FileGroupId] int NOT NULL,
    TotalExtents int NOT NULL,
    UsedExtents int NOT NULL,
    [FileName] sysname NOT NULL,
```

```
[FilePath] nvarchar(MAX) NOT NULL,  
[FileGroup] varchar(MAX) NULL  
  
create table #Results(  
    db sysname NULL ,  
    FileType varchar(4) NOT NULL,  
    [FileGroup] sysname not null,  
    [FileName] sysname NOT NULL,  
    TotalMB numeric(18,2) NOT NULL,  
    UsedMB numeric(18,2) NOT NULL,  
    PctUsed numeric(18,2) NULL,  
    FilePath nvarchar(MAX) NULL,  
    FileID int null)  
  
create table #Log(  
    db sysname NOT NULL,  
    LogSize numeric(18,5) NOT NULL,  
    LogUsed numeric(18,5) NOT NULL,  
    Status int NOT NULL,  
    [FilePath] nvarchar(MAX) NULL)  
  
INSERT #Data (FileID, [FileGroupId], TotalExtents, UsedExtents,  
[FileName], [FilePath])  
EXEC ('DBCC showfilestats WITH NO_INFOMSGS')  
  
update #Data  
set #Data.FileGroup = sysfilegroups.groupname  
from #Data, sysfilegroups  
where #Data.FileGroupId = sysfilegroups.groupid  
  
INSERT INTO #Results (db, [FileGroup], FileType, [FileName],  
TotalMB, UsedMB, PctUsed, FilePath, FileID)  
SELECT DB_NAME() db,  
    [FileGroup],  
    'Data' FileType,  
    [FileName],  
    TotalExtents * 64./1024. TotalMB,  
    UsedExtents *64./1024 UsedMB,  
    UsedExtents*100. /TotalExtents UsedPct,  
    [FilePath],  
    FileID  
FROM #Data  
order BY --1,2  
DB_NAME(), [FileGroup]
```

SQL 2014 - Módulo III

```
insert #Log (db, LogSize, LogUsed, Status)
exec('dbcc sqlperf(logspace) WITH NO_INFOMSGS ')

insert #Results(db, [FileGroup], FileType, [FileName],
TotalMB, UsedMB, PctUsed, FilePath, FileID)
select DB_NAME() db,
       'Log' [FileGroup],
       'Log' FileType,
       s.[name] [FileName],
       s.Size/128. as LogSize ,
       FILEPROPERTY(s.name,'spaceused')/8.00 /16.00 As
LogUsedSpace,
       ((FILEPROPERTY(s.name,'spaceused')/8.00 /16.00)*100)/
(s.Size/128.) UsedPct,
       s.FileName FilePath,
       s.FileID FileID
from #Log l , master.dbo.sysaltfiles f , dbo.sysfiles s
where f.dbid = DB_ID()
and (s.status & 0x40) <> 0
and s.FileID = f.FileID
and l.db = DB_NAME()

SELECT r.db AS "Database",
r.FileType AS "File type",
CASE
    WHEN r.FileGroup = 'Log' Then 'N/A'
    ELSE r.FileGroup
END "File group",
r.FileName AS "Logical file name",
r.TotalMB AS "Total size (MB)",
r.UsedMB AS "Used (MB)",
r.PctUsed AS "Used (%)",
r.FilePath AS "File name",
r.FileID AS "File ID",
CASE WHEN s.maxsize = -1 THEN null
    ELSE CONVERT(decimal(18,2), s.maxsize /128.)
END "Max. size (MB)",
CONVERT(decimal(18,2), s.growth /128.) "Autogrowth increment
(MB)"
FROM #Results r
INNER JOIN dbo.sysfiles s
ON r.FileID = s.FileID
ORDER BY 1,2,3,4,5
```

```
DROP TABLE #Data  
DROP TABLE #Results  
DROP TABLE #Log
```

Para verificar as esperas por I/O (I/O waits), use este código:

```
SELECT DB_NAME(database_id) AS [Database Name] ,  
       file_id AS [File ID],  
       io_stall_read_ms AS [Total Read Waits (ms)],  
       num_of_reads AS [Number of Reads],  
       CAST(io_stall_read_ms / ( 1.0 + num_of_reads ) AS  
NUMERIC(10, 1))  
              AS [Average Read Wait (ms)],  
       io_stall_write_ms AS [Total Write Waits (ms)],  
       num_of_writes AS [Number of Writes],  
       CAST(io_stall_write_ms / ( 1.0 + num_of_writes ) AS  
NUMERIC(10, 1))  
              AS [Average Write Wait (ms)],  
       io_stall_read_ms + io_stall_write_ms AS [Total I/O Waits  
(ms)] ,  
       num_of_reads + num_of_writes AS [Number of I/O  
Operations],  
       CAST(( io_stall_read_ms + io_stall_write_ms ) / ( 1.0 +  
num_of_reads  
              + num_  
of_writes)  
              AS NUMERIC(10,1)) AS [Average I/O Wait (ms)]  
FROM sys.dm_io_virtual_file_stats(NULL, NULL)  
ORDER BY [Average I/O Wait (ms)] DESC ;
```

Para obter o tamanho de todos os bancos de dados de uma instância, utilize este código:

```
if exists (select * from tempdb.sys.all_objects where name like  
'%'#dbsize%')  
drop table #dbsize  
create table #dbsize  
(Dbname varchar(30), dbstatus varchar(20), Recovery_Model  
varchar(10) default ('NA'), file_Size_MB decimal(20,2) default  
(0), Space_Used_MB decimal(20,2) default (0), Free_Space_MB  
decimal(20,2) default (0))  
go
```

SQL 2014 - Módulo III

```
insert into #dbsize(Dbname, dbstatus, Recovery_Model, file_Size_
MB, Space_Used_MB, Free_Space_MB)
exec sp_msforeachdb
`use [?];
select DB_NAME() AS DbName,
CONVERT(varchar(20), DatabasePropertyEx('?' , 'Status')) ,
CONVERT(varchar(20), DatabasePropertyEx('?' , 'Recovery')) ,
sum(size)/128.0 AS File_Size_MB,
sum(CAST(FILEPROPERTY(name, 'SpaceUsed') AS INT))/128.0 as
Space_Used_MB,
SUM( size)/128.0 - sum(CAST(FILEPROPERTY(name, 'SpaceUsed') AS
INT))/128.0 AS Free_Space_MB
from sys.database_files where type=0 group by type'
```

Este código mostra o espaço ocupado no transaction log:

```
SELECT db.[name] AS [Database Name] ,
db.recovery_model_desc AS [Recovery Model] ,
db.log_reuse_wait_desc AS [Log Reuse Wait Description] ,
ls.cntr_value AS [Log Size (KB)] ,
lu.cntr_value AS [Log Used (KB)] ,
CAST(CAST(lu.cntr_value AS FLOAT) / CAST(ls.cntr_value AS FLOAT)
AS DECIMAL(18,2)) * 100 AS [Log Used %] ,
db.[compatibility_level] AS [DB Compatibility Level] ,
db.page_verify_option_desc AS [Page Verify Option]
FROM sys.databases AS db
INNER JOIN sys.dm_os_performance_counters AS lu
ON db.name = lu.instance_name
INNER JOIN sys.dm_os_performance_counters AS ls
ON db.name = ls.instance_name
WHERE lu.counter_name LIKE 'Log File(s) Used Size (KB)%'
AND ls.counter_name LIKE 'Log File(s) Size (KB)%' ;
```

Já este código mostra se é necessária a reconstrução de índices:

```
-- especifica o database desejado
USE [master] ;
GO
--IF EXISTS (SELECT 1 FROM tempdb.sys .tables WHERE [name] like
`#tbllist%`)
--      DROP TABLE #tbllist;
```

```
CREATE TABLE #tbllist(FullName varchar (255) NOT NULL,
DatabaseName varchar(255), TableName varchar (255) NOT NULL)
GO

-- obtém as partes dos nomes dos arquivos
sp_msforeachdb
`INSERT INTO #tbllist SELECT ''[`` + ``?`` + ``].[`` + [TABLE_
SCHEMA] + ``].[`` + [TABLE_NAME] + ``]`` as FullName, ``[`` + ``?``
+ ``]`` as DatabaseName, [TABLE_NAME] as TableName
FROM [?].INFORMATION_SCHEMA.TABLES
WHERE TABLE_CATALOG <> ``tempdb`` AND TABLE_TYPE = ``BASE
TABLE````

-- declara variáveis
SET NOCOUNT ON ;
DECLARE @fullname VARCHAR (255);
DECLARE @DatabaseName varchar (255);
DECLARE @tableName varchar (255);
DECLARE @execstr VARCHAR (255);
DECLARE @objectid INT ;
DECLARE @indexid INT ;
DECLARE @frag decimal ;
DECLARE @maxfrag decimal ;

-- define o máximo de fragmentação.
SET @maxfrag = 30.0 ;

-- declara o cursor.
DECLARE tables CURSOR FOR
SELECT FullName, DatabaseName, TableName
FROM #tbllist

-- cria a tabela.
CREATE TABLE #fraglist (
    ObjectName varchar (255),
    ObjectId INT ,
    IndexName varchar (255),
    IndexId INT ,
    Lvl INT ,
    CountPages INT ,
    CountRows INT ,
    MinRecSize INT ,
```

SQL 2014 - Módulo III

```
    MaxRecSize INT ,
    AvgRecSize INT ,
    ForRecCount INT ,
    Extents INT ,
    ExtentSwitches INT ,
    AvgFreeBytes INT ,
    AvgPageDensity INT ,
    ScanDensity decimal ,
    BestCount INT ,
    ActualCount INT ,
    LogicalFrag decimal ,
    ExtentFrag decimal );
-- abre o cursor.
OPEN tables;
-- realiza um loop para todas as tabelas do banco de dados
FETCH NEXT
FROM tables
INTO @fullname, @DatabaseName, @tableName;
WHILE @@FETCH_STATUS = 0
BEGIN;

-- execute DBCC SHOWCONTIG para todos os índices
INSERT INTO #fraglist
    EXEC ('DBCC SHOWCONTIG ('' + @fullname + '')'
        WITH FAST, TABLERESULTS, ALL_INDEXES, NO_INFOMSGS' );
--put the full table name into the object name, as we need it
below
UPDATE #fraglist
SET ObjectName = @fullname
WHERE ObjectName = @tableName

FETCH NEXT
FROM tables
INTO @fullname, @DatabaseName, @tableName;
END;

-- Fecha o cursor e o desaloca.
CLOSE tables ;
DEALLOCATE tables ;
```

```

SELECT DISTINCT
    IDENTITY(int ,1, 1) as ord
    , `Executing USE ` + T.DatabaseName + `; ` +
    `IF ((SELECT INDEXPROPERTY (` + CAST( F.ObjectId as
varchar(255 )) + `, ` + CHAR(39 ) + F. IndexName + CHAR( 39) + `,
`IndexDepth'')) > 0) ` +
        `ALTER INDEX ` + RTRIM(F.IndexName) + ` ON ` + RTRIM (T.
FullName) + ` REBUILD - ` + CAST(LogicalFrag as varchar(5)) + `%`%
Fragmented` as [task_descr]
    , `USE ` + T.DatabaseName + `; ` +
    `IF ((SELECT INDEXPROPERTY (` + CAST( F.ObjectId as
varchar(255 )) + `, ` + CHAR(39 ) + F. IndexName + CHAR( 39) + `,
`IndexDepth'')) > 0) ` +
        `ALTER INDEX [ ` + RTRIM(F.IndexName) + `] ON ` + RTRIM
(T. FullName) + ` REBUILD` as [exec_sql]
INTO #tmp_exec_rebuild_index
FROM #fraglist as F
    INNER JOIN #tbllist as T ON T.FullName = f.ObjectName
WHERE LogicalFrag >= @maxfrag
ORDER BY 1

DECLARE @max_loop int,
@loopcount int,
@exec_sql varchar (4000),
@exec_descr varchar (4000)

SET @max_loop = (SELECT MAX([ord]) FROM #tmp_exec_rebuild_index)
SET @loopcount = 1

WHILE (@loopcount <=@max_loop )
BEGIN
    SET @exec_descr = (SELECT [task_descr] FROM #tmp_exec_
rebuild_index WHERE [ord] = @loopcount)
    SET @exec_sql = (SELECT [exec_sql] FROM #tmp_exec_
rebuild_index WHERE [ord] = @loopcount )
    PRINT @exec_descr
    EXEC(@exec_sql );
    SET @loopcount = @loopcount + 1
END

```

SQL 2014 - Módulo III

```
-- exclui as tabelas temporárias.  
DROP TABLE #fraglist ;  
DROP TABLE #tbllist ;  
DROP TABLE #tmp_exec_rebuild_index  
GO
```

Este código exibe a lista de índices não utilizados:

```
SELECT OBJECT_NAME(i.[object_id]) AS [Table name] ,  
CASE WHEN i.name IS NULL THEN '<Unused table>' ELSE i.name END  
AS [Index name]  
FROM sys.indexes AS i  
INNER JOIN sys.objects AS o ON i.[object_id] = o.[object_id]  
WHERE i.index_id NOT IN (  
    SELECT s.index_id  
    FROM sys.dm_db_index_usage_stats AS s  
    WHERE s.[object_id] = i.[object_id]  
    AND i.index_id = s.index_id  
    AND database_id = DB_ID()  
    AND o.[type] = 'U'  
) ORDER BY OBJECT_NAME(i.[object_id]) ASC;
```

Por meio deste código, podemos obter a quantidade de páginas alocadas em memória por cada banco de dados:

```
SELECT COUNT(*) AS cached_pages_count  
, CASE database_id  
    WHEN 32767 THEN 'ResourceDb'  
    ELSE db_name(database_id)  
END AS database_name  
FROM sys.dm_os_buffer_descriptors  
GROUP BY db_name(database_id), database_id  
ORDER BY cached_pages_count DESC;
```

Para obter a taxa de performance do SQL Server 2014, utilizamos o seguinte código:

```
DECLARE @PERF_LARGE_RAW_FRACTION INT ,  
        @PERF_LARGE_RAW_BASE INT  
SELECT @PERF_LARGE_RAW_FRACTION = 537003264 ,  
        @PERF_LARGE_RAW_BASE = 1073939712  
  
SELECT dopc_fraction.object_name AS [Performance object],
```

```
dopc_fraction.instance_name AS [Counter instance],  
dopc_fraction.counter_name AS [Counter name],  
CONVERT(DECIMAL(38,2), CAST(dopc_fraction.cntr_value AS  
FLOAT)  
/ CAST(CASE dopc_base.cntr_value  
WHEN 0 THEN NULL  
ELSE dopc_base.cntr_value  
END AS FLOAT)) AS [Value]  
FROM sys.dm_os_performance_counters AS dopc_base  
JOIN sys.dm_os_performance_counters AS dopc_fraction  
ON dopc_base.cntr_type = @PERF_LARGE_RAW_BASE  
AND dopc_fraction.cntr_type = @PERF_LARGE_RAW_  
FRACTION  
AND dopc_base.object_name = dopc_fraction.object_  
name  
AND dopc_base.instance_name = dopc_fraction.  
instance_name  
AND ( REPLACE(UPPER(dopc_base.counter_name),  
'BASE', '') =  
      UPPER(dopc_fraction.counter_name)  
OR REPLACE(UPPER(dopc_base.counter_name), 'BASE',  
'') =  
      REPLACE(UPPER(dopc_fraction.counter_name),  
'RATIO', '')  
 )  
ORDER BY dopc_fraction.object_name ,  
dopc_fraction.instance_name ,  
dopc_fraction.counter_name
```

Para testar conexões entre servidores SQL Server, utilize este código:

```
Create procedure usp_lsping(@nf nchar = 'N', @wtn nchar = 'E')  
WITH ENCRYPTION  
as  
set nocount on  
BEGIN  
DECLARE @LSrvrs TABLE  
(  
SrvrID int IDENTITY(1,1) PRIMARY KEY,  
SrvName nvarchar(128)  
)  
  
insert into @LSrvrs  
select srvname from sys.sysservers
```

SQL 2014 - Módulo III

```
where srvname != CONVERT(nvarchar(128),
 SERVERPROPERTY('servername'));
;

-- declare @maxloocnt int;
declare @loopcnt int;
declare @srvr nvarchar(128);
declare @retval int;
declare @Clr nvarchar(4)
declare @msg nvarchar(MAX)
declare @errchk int;
--

set @errchk = 0;
set @Clr = char(13)+char(10);
select @maxloocnt = count(*) from @LSrvrs;
set @loopcnt = 1;
set @msg = '';
set @msg = @msg + '<Start>' +@Clr
while @loopcnt <= @maxloocnt
begin
select @srvr = srvname from @LSrvrs where SrvrID = @loopcnt;
--select @srvr
begin try
exec @retval = sys.sp_testlinkedserver @srvr;
end try
begin catch
set @retval = sign(@@error);
set @errchk = 99;
end catch;
--if @retval =qwqw 0
set @msg = @msg + @Clr+@srvr + ':' + cast(@retval as
nvarchar(100));
set @loopcnt = @loopcnt + 1;
end;
set @msg = @msg + @Clr+'<end>' +@Clr
if @nf = 'Y'
begin
--if @wtn = 'E' or @wtn = 'S' or @wtn = 'B'
--BEGIN
--print 'WTN:' +@wtn
--print '@errchk:' + CONVERT(nvarchar(128),@errchk)
if (@wtn = 'S' and @errchk = 0) or (@wtn = 'E' and @errchk > 0)
or(@wtn = 'B' )
BEGIN
```

```
--select @msg;
EXEC msdb.dbo.sp_send_dbmail
@recipients=N'altere para informar o seu email',@body=@msg,
@subject ='Linked Server Ping'--,@query ='select @msg;',
--@attach_query_result_as_file = 1,@query_attachment_filename
='PingResults.txt'
END;
--else
-- BEGIN
--
-- END
END
--end
--else
--BEGIN
-- '';
--END;

END
set nocount off
```

Para realizar a leitura do errorlog, sendo o primeiro parâmetro o número do errorlog e o segundo o tipo de log a ser lido (1: errorlog da instância e 2: errorlog do SQL Server Agent), utilize este código:

```
EXEC master.dbo.xp_readerrorlog 0, 1, NULL, NULL, NULL, NULL,
N'desc'
```

Veja como forçar a troca de errorlog:

```
EXEC sp_cycle_errorlog
```

O seguinte código é utilizado para determinar a utilização de CPU:

```
DECLARE @ts_now AS BIGINT
SELECT @ts_now = cpu_ticks / ( cpu_ticks / ms_ticks ) FROM sys.
dm_os_sys_info

SELECT TOP (@span)
    SQLProcessUtilization AS [SQL Server CPU Utilization (%)]
    ,
    SystemIdle AS [System Idle Process (%)] ,
    100 - SystemIdle - SQLProcessUtilization
```

SQL 2014 - Módulo III

```
AS [Other Process CPU Utilization
(%)] ,
    DATEADD(ms, -1 * ( @ts_now - [timestamp] ), GETDATE())
        AS [Event Time]
FROM      ( SELECT      record.value('(../Record/@id)[1]', 'int') AS
record_id ,
                    record.value('(../Record/
SchedulerMonitorEvent/
SystemHealth/SystemIdle)
[1]', 'int')
                AS
[SystemIdle] ,
                    record.value('(../Record/
SchedulerMonitorEvent/
SystemHealth/
ProcessUtilization)[1]',
                    'int')
                AS
[SQLProcessUtilization] ,
                    [timestamp]
FROM      ( SELECT      [timestamp] ,
CONVERT(XML, record) AS [record]
sys.dm_os_ring_buffers
ring_buffer_type =
N'RING_BUFFER_SCHEDULER_
MONITOR'
                    AND record LIKE
N'%<SystemHealth>%'
                ) AS x
            ) AS y
        ORDER BY record_id DESC ;
```

Este código permite definir os dez comandos SQL que mais consomem recursos:

```
use tempdb

go

IF object_id('tempdb..##FindTopCPUQueries_set1') is not null DROP
TABLE [dbo].[##FindTopCPUQueries_set1]

GO

declare @ServerTime datetime = getdate()
```

```
, @ConvertMiliSeconds bigint = 1000  
  
, @FilterMoreThanMiliSeconds bigint = 1  
  
, @FilterHours bigint = 2  
  
, @execution_count bigint = 2  
  
, @debugFlg bit = 0  
  
if @debugFlg=1 select @ServerTime as ServerTime, @  
ConvertMiliSeconds as ConvertMiliSeconds  
  
, @FilterMoreThanMiliSeconds as FilterMoreThanMiliSeconds, @  
FilterHours as FilterHours  
  
, @execution_count as execution_count  
  
select TOP 10  
@@servername as servername, @ServerTime as runtime  
, isnull(db_name(QueryText.dbid), 'PreparedSQL') as DBName  
, SUBSTRING(QueryText.text, (QueryStats.statement_start_  
offset/2)+1,  
(isnull((  
CASE QueryStats.statement_end_offset  
WHEN -1 THEN DATALENGTH(QueryText.text)  
WHEN 0 THEN DATALENGTH(QueryText.text)  
ELSE QueryStats.statement_end_offset
```

SQL 2014 - Módulo III

```
END - QueryStats.statement_start_offset),0)/2)

+ 1) AS QueryExecuted

, total_worker_time AS total_worker_time

,QueryStats.execution_count as execution_count

,statement_start_offset,statement_end_offset

,(case when QueryText.dbid is null then OBJECT_NAME(QueryText.
objectid) else OBJECT_NAME(QueryText.objectid, QueryText.dbid)
end) as ObjectName

,query_hash

,plan_handle

,sql_handle

into ##FindTopCPUQueries_set1

from sys.dm_exec_query_stats as QueryStats

cross apply sys.dm_exec_sql_text(QueryStats.sql_handle) as
QueryText

where QueryStats.query_hash IN

(
select QueryStatsBaseTable.query_hash

from sys.dm_exec_query_stats QueryStatsBaseTable

group by query_hash

)
```

```
ORDER BY total_worker_time/execution_count DESC;

if @debugFlg=1 select * from ##FindTopCPUQueries_set1 order by
QueryExecuted

IF object_id('tempdb..##FindTopCPUQueries_set2') is not null DROP
TABLE [dbo].[##FindTopCPUQueries_set2]

select

servername, runtime, max(DBName) as DBName, max(QueryExecuted) as
QueryExecuted, (sum(total_worker_time)/sum(execution_count)) /@
ConvertMiliSeconds as AvgCPUTime

,sum(execution_count) as execution_count, query_hash,
max(ObjectName) as ObjectName

into ##FindTopCPUQueries_set2

from ##FindTopCPUQueries_set1

group by query_hash, servername, runtime

order by AvgCPUTime desc

select * from ##FindTopCPUQueries_set2

--where QueryExecuted like 'select TOP 300%'

order by AvgCPUTime desc
```

Já este código mostra os comandos que mais consomem I/O:

```
SELECT TOP 10
[Total IO] = (qs.total_logical_reads + qs.total_logical_writes)
, [Average IO] = (qs.total_logical_reads + qs.total_logical_
writes) /
qs.execution_count
, qs.execution_count
, SUBSTRING (qt.text, (qs.statement_start_offset/2) + 1,
```

SQL 2014 - Módulo III

```
( (CASE WHEN qs.statement_end_offset = -1
THEN LEN(CONVERT(NVARCHAR(MAX), qt.text)) * 2
ELSE qs.statement_end_offset
END - qs.statement_start_offset)/2) + 1) AS [Individual Query]
, qt.text AS [Parent Query]
, DB_NAME(qt.dbid) AS DatabaseName
, qp.query_plan
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) as qt
CROSS APPLY sys.dm_exec_query_plan(qs.plan_handle) qp
ORDER BY [Total IO] DESC
```

Veja como obter os recursos que sofrem mais espera (WAITS):

```
WITH Waits
      AS ( SELECT      wait_type ,
                        wait_time_ms / 1000. AS wait_time_sec ,
                        100. * wait_time_ms / SUM(wait_time_ms)
OVER ( ) AS pct ,
ROW_NUMBER() OVER ( ORDER BY wait_time_ms
DESC ) AS rn
      FROM sys.dm_os_wait_stats
      WHERE wait_type NOT IN ( 'CLR_SEMAPHORE',
'LAZYWRITER_SLEEP',
'SLEEP_TASK',
'SLEEP_SYSTEMTASK',
'SQLTRACE_BUFFER_',
'LOGMGR_QUEUE',
'FLUSH', 'WAITFOR',
'CHECKPOINT_QUEUE' )
      )
      SELECT      wait_type AS [Wait Type],
                  CAST(wait_time_sec AS DECIMAL(12, 2)) AS [Wait Time
(s)] ,
                  CAST(pct AS DECIMAL(12, 2)) AS [Wait Time (%) ]
      FROM Waits
      WHERE pct > 10
```

Para verificar a memória disponível no servidor e a memória disponível para o SQL Server, utilize o código a seguir:

```
DECLARE  
@TotalMEMORYinBytes NUMERIC,  
@TotalMEMORYinMegaBytes NUMERIC,  
@SQLMaxMemoryMegaByte NUMERIC,  
@RamOver16GB NUMERIC --Usado para testar memória superior a 16gb  
  
-- Buscando a memória física do servidor  
SET @TotalMEMORYinBytes = (select (physical_memory_kb + virtual_  
memory_kb) * 1024 from sys.dm_os_sys_info)  
  
-- Convertendo bytes em mb  
SET @TotalMEMORYinMegaBytes = (@TotalMEMORYinBytes / (1024*1024))  
  
-- Sistema Operacional necessita de pelo menos 1Gb  
SET @SQLMaxMemoryMegaByte = 1024  
  
-- Se o total é maior que 16gb, adicionar 4Gb  
IF @TotalMEMORYinMegaBytes > 16384  
BEGIN  
    SET @SQLMaxMemoryMegaByte = (@SQLMaxMemoryMegaByte + 4096) SET @  
    RamOver16GB = ((@TotalMEMORYinMegaBytes - 16384)/8) SET @  
    SQLMaxMemoryMegaByte = (@SQLMaxMemoryMegaByte + @RamOver16GB)  
END  
  
-- Verifica se o total está em 12 e 16Gb  
IF (@TotalMEMORYinMegaBytes < 16384 and @TotalMEMORYinMegaBytes >  
12288 ) SET @SQLMaxMemoryMegaByte = (@SQLMaxMemoryMegaByte + 4 )  
  
-- Verifica se o total está entre 8 e 12gb  
IF (@TotalMEMORYinMegaBytes < 12288 and @TotalMEMORYinMegaBytes >  
8192) SET @SQLMaxMemoryMegaByte = (@SQLMaxMemoryMegaByte + 3 )  
  
-- Verifica se o total está entre 4 e 8Gb  
IF (@TotalMEMORYinMegaBytes < 8192 and @TotalMEMORYinMegaBytes >  
4096) SET @SQLMaxMemoryMegaByte = (@SQLMaxMemoryMegaByte + 2 )  
  
-- Verifica se o total é menor que 4Gb  
  
IF @TotalMEMORYinMegaBytes < 4096 SET @SQLMaxMemoryMegaByte = (@  
SQLMaxMemoryMegaByte + 0 )
```

SQL 2014 - Módulo III

```
-- Calcula o máximo de memória em MB  
SET @SQLMaxMemoryMegaByte = (@TotalMEMORYinMegabytes - @  
SQLMaxMemoryMegaByte)  
  
-- Mostra o valor total de memória disponível  
SELECT @TotalMEMORYinMegabytes /1024 AS 'Memória Servidor em  
MB****', @SQLMaxMemoryMegaByte /1024 AS 'Memória SQL Server em MB  
****'  
GO
```

Agora, veja como utilizar o database buffer cache:

```
DECLARE @total_buffer INT;  
  
SELECT @total_buffer = cntr_value  
    FROM sys.dm_os_performance_counters  
   WHERE RTRIM([object_name]) LIKE '%Buffer Manager'  
     AND counter_name = 'Total Pages';  
  
WITH src AS  
(  
    SELECT  
        database_id, db_buffer_pages = COUNT_BIG(*)  
      FROM sys.dm_os_buffer_descriptors  
     --WHERE database_id BETWEEN 5 AND 32766  
      GROUP BY database_id  
)  
SELECT  
    [db_name] = CASE [database_id] WHEN 32767  
        THEN 'Resource DB'  
        ELSE DB_NAME([database_id]) END,  
    db_buffer_pages,  
    db_buffer_MB = db_buffer_pages / 128,  
    db_buffer_percent = CONVERT(DECIMAL(6,3),  
        db_buffer_pages * 100.0 / @total_buffer)  
  FROM src  
 ORDER BY db_buffer_MB DESC;
```

Por último, o seguinte código é utilizado para listar os comandos SQL que estão aguardando por I/O:

```
SELECT
    s.session_id [Sessao],
    s.host_name [Origem],
    r.last_wait_type [TipoEspera],
    C.local_net_address [Destino],
    c.last_read [Ult. Leitura],
    c.last_write[Ult. Escrita],
    T.text
FROM sys.dm_exec_connections c
CROSS APPLY sys.dm_exec_sql_text(most_recent_sql_handle) AS T
INNER JOIN sys.dm_exec_sessions s
ON s.session_id = c.session_id
INNER JOIN sys.dm_exec_requests r
ON s.session_id = r.session_id
WHERE r.last_wait_type = 'ASYNC_NETWORK_IO'
```

Fernando
357.907.3352

Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo.

- A monitoração de ambiente é fundamental para um ambiente de banco de dados bem gerenciado;
- É fundamental dividir a monitoração para poder controlar seus pontos de forma organizada;
- É importante ter um bom conjunto de scripts para monitorar todas as áreas do banco de dados: Memória, Processador, I/O, Contenções/Esperas;
- O administrador precisa desenvolver controles para atividades a serem executadas diariamente, semanalmente e mensalmente, sejam manuais ou automáticas através de tarefas;
- A auditoria do banco de dados pode ajudar a determinar problemas de performance junto ao banco de dados e a buscar problemas;
- A segurança em um banco de dados depende de vários aspectos, como controle, monitoração e backups.

Gerenciando um banco de dados

10

Teste seus conhecimentos

Fernando Sposito
357.907.



IMPACTA
EDITORA

1. Com relação ao monitoramento dos backups realizados, qual a alternativa correta?

- a) Ele é feito para garantir que o backup está sendo realizado com sucesso.
- b) O SQL é confiável e não é necessário o monitoramento.
- c) Ele é feito para gerar informações estatísticas.
- d) Não é necessário, pois o SQL envia uma mensagem de erro.
- e) Nenhuma das alternativas anteriores está correta.

2. Com relação à auditoria, qual a alternativa incorreta?

- a) A auditoria é importante para controlar ações do banco.
- b) Podemos utilizar auditoria para monitorar ações indevidas.
- c) Podemos auditar dados.
- d) A auditoria pode ser realizada para a INSTANCE.
- e) Prejudica a performance e não deve ser utilizada.

3. Com relação ao monitoramento, qual a alternativa incorreta?

- a) O monitoramento garante um bom funcionamento do ambiente.
- b) Podemos monitorar memória, processador, I/O, contenção e backup.
- c) O monitoramento deve ser diário, semanal e mensal.
- d) Podemos evitar problemas antes de acontecer.
- e) Não é necessário, pois o SQL possui alta disponibilidade.

4. Como podemos monitorar a alteração de objetos em um banco de dados?

- a) Através de auditoria do SQL.
- b) Auditoria do SQL e gatilhos do tipo DDL.
- c) Através de gatilhos do tipo DDL.
- d) Não é necessário, pois ocupa muito espaço em disco.
- e) Nenhuma das alternativas anteriores está correta.

5. Qual alternativa não corresponde a uma classificação de auditoria?

- a) Objetos
- b) Tabelas
- c) Segurança
- d) Instância/Servidor
- e) Dados

10

Gerenciando um banco de dados

Mãos à obra!

Fernando Spinola
357.900-088



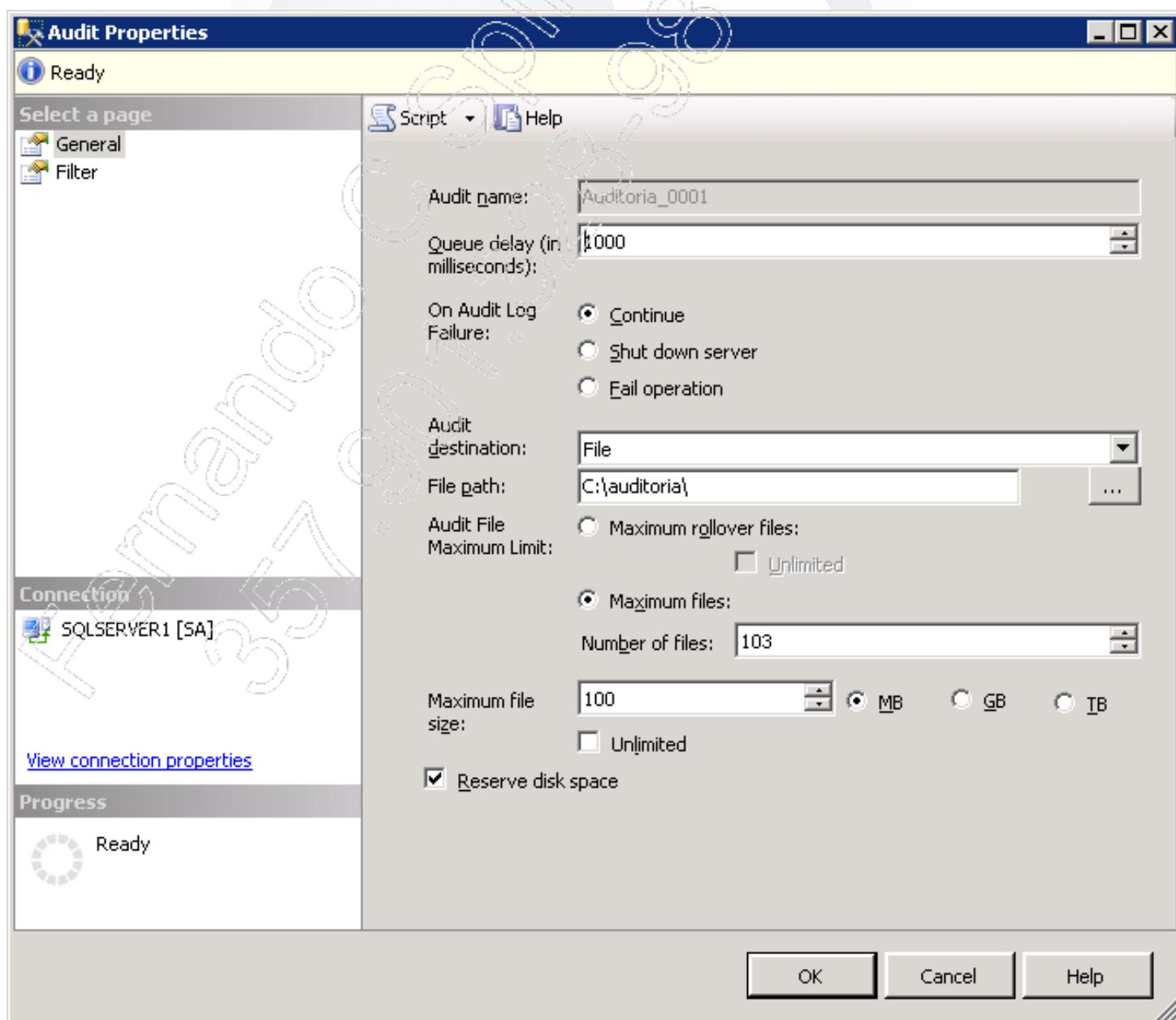
IMPACTA
EDITORA

Laboratório 1

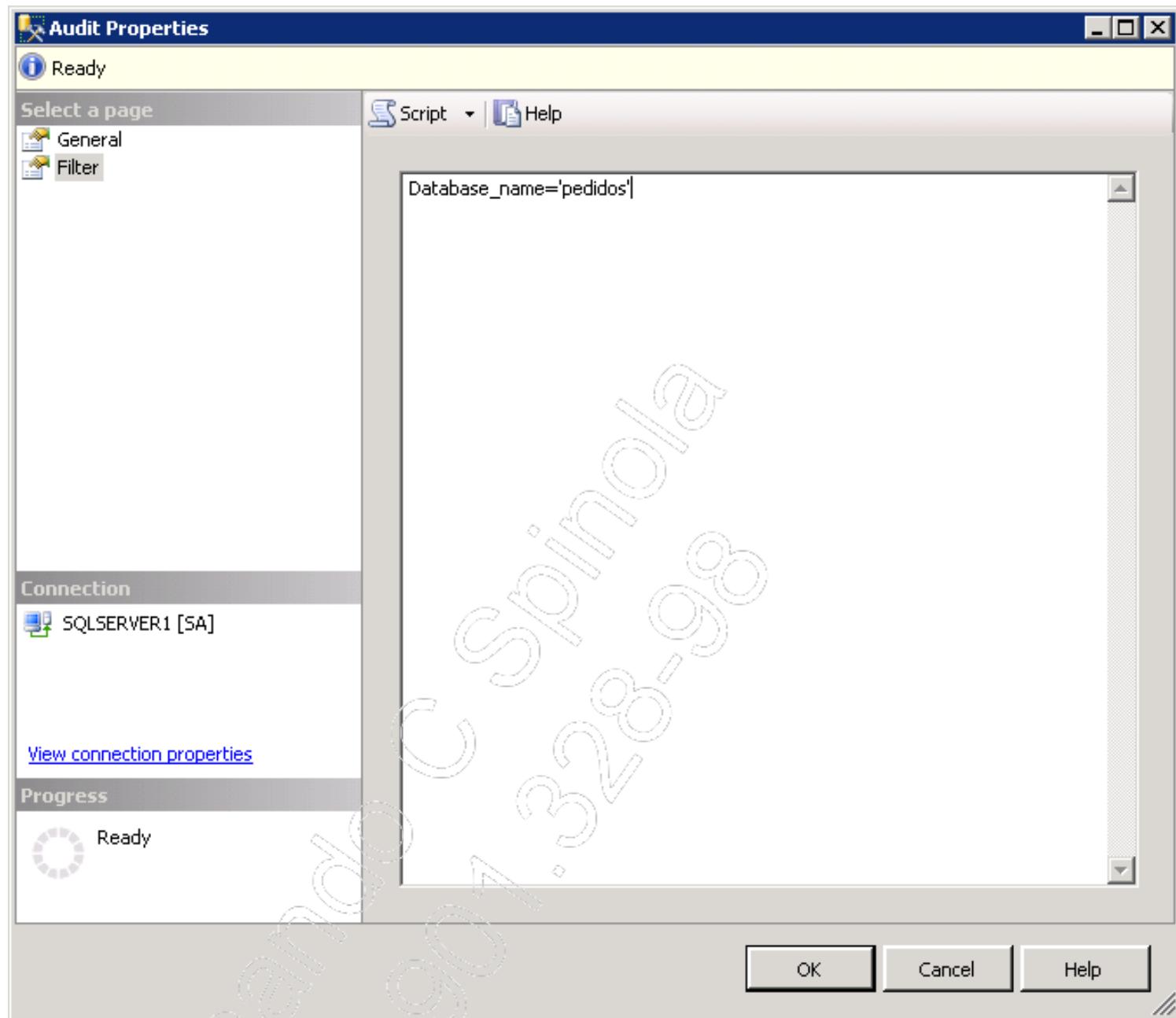
A – Fazendo auditoria

Neste laboratório, vamos realizar uma das atividades de um administrador de banco de dados: a auditoria. Veremos a auditoria de servidor (Server) e a de banco de dados (Database). Para isso, devemos seguir os passos adiante:

1. Crie o diretório **c:\auditoria**;
2. Selecione a pasta **Security**, em seguida, clique em **Audits** com o botão direito do mouse e selecione a opção **New Audit**;
3. Em **General**, aplique as seguintes configurações:



4. Selecione a opção **Filter** e digite **Database_name='pedidos'** no campo em branco. Em seguida, clique no botão **OK**:



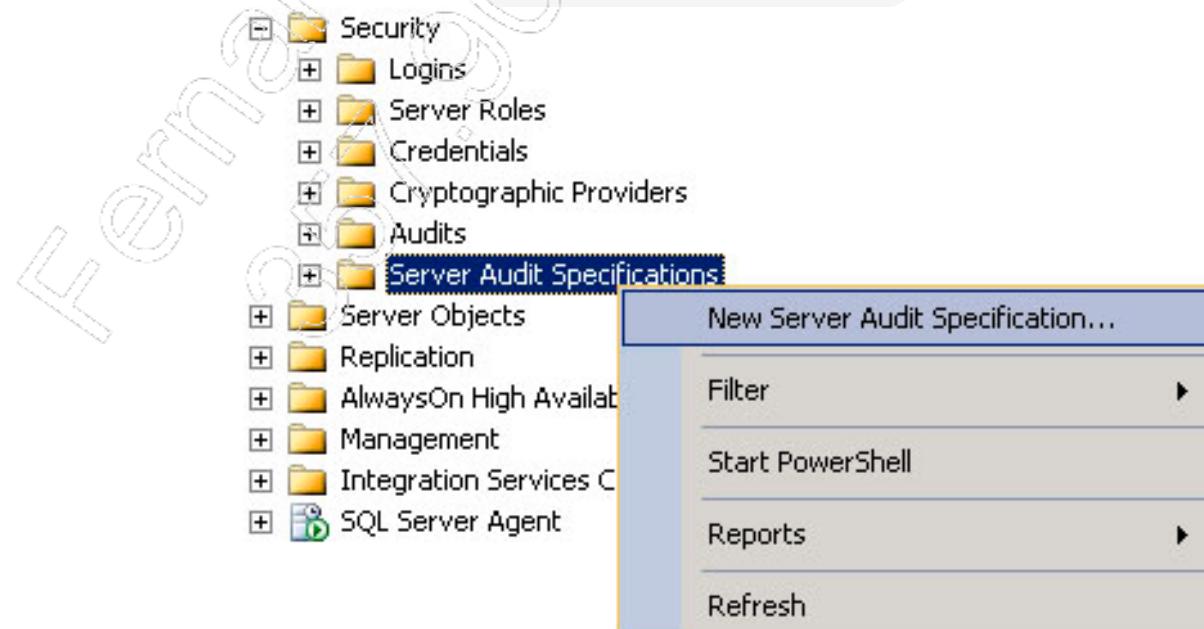
SQL 2014 - Módulo III

Outra opção para criar o Audit é digitar o código adiante:

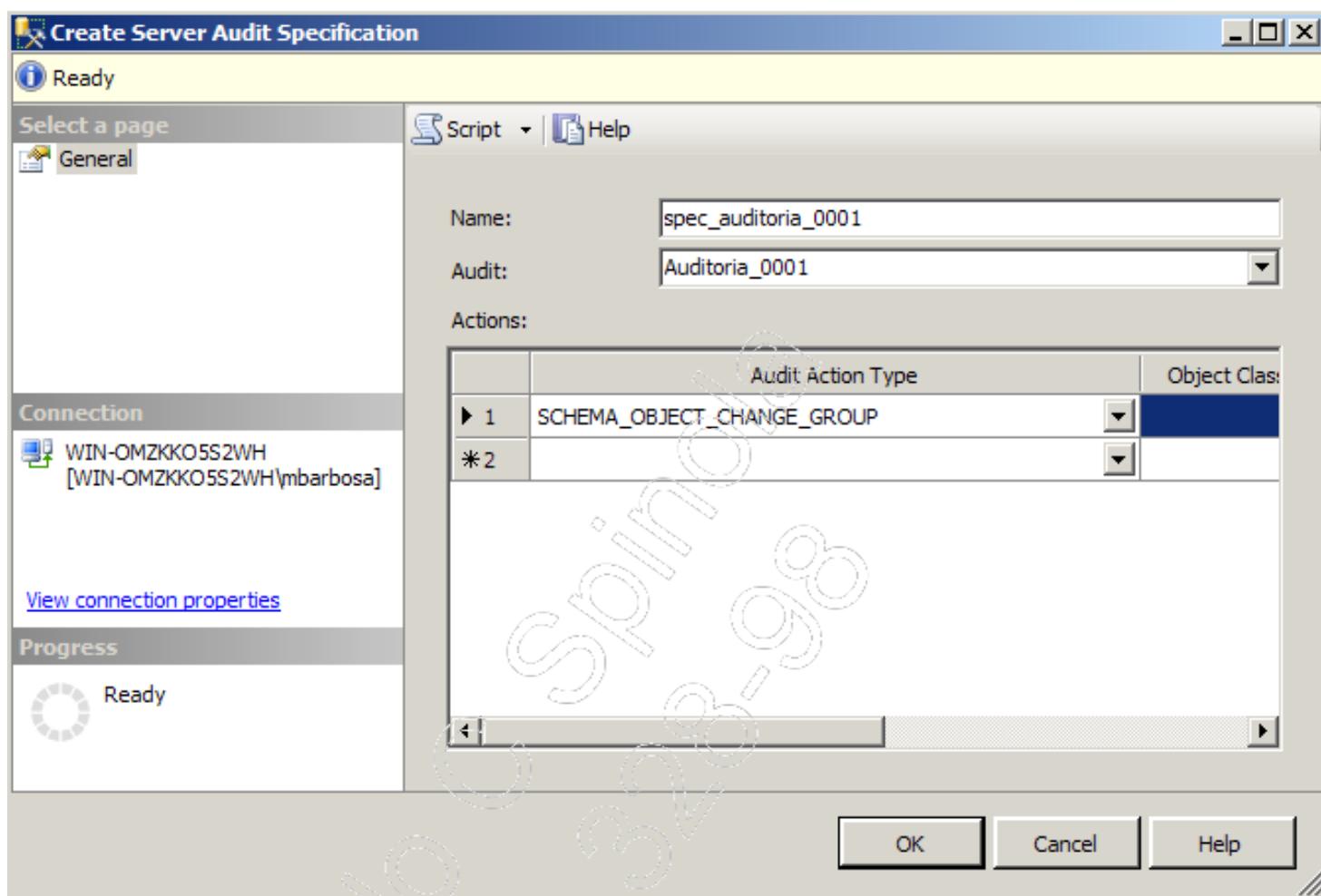
```
USE [master]
GO

CREATE SERVER AUDIT [Auditoria_0001]
TO FILE
(
    FILEPATH = N'c:\auditoria'
    ,MAXSIZE = 100 MB
    ,MAX_FILES = 103
    ,RESERVE_DISK_SPACE = ON
)
WITH
(
    QUEUE_DELAY = 1000
    ,ON_FAILURE = CONTINUE
)
WHERE Database_name='pedidos'
GO
```

5. No Object Explorer, selecione a pasta Security e a opção Server Audit Specification;
6. Com o botão direito do mouse, selecione a opção New Server Audit Specification...:



7. Informe o nome e o tipo da ação de auditoria e indique o nome da auditoria vinculada como mostra a imagem a seguir. Em seguida clique em **OK**:



Outra opção para criar a especificação da auditoria é utilizar o script a seguir:

```
USE [master]
GO

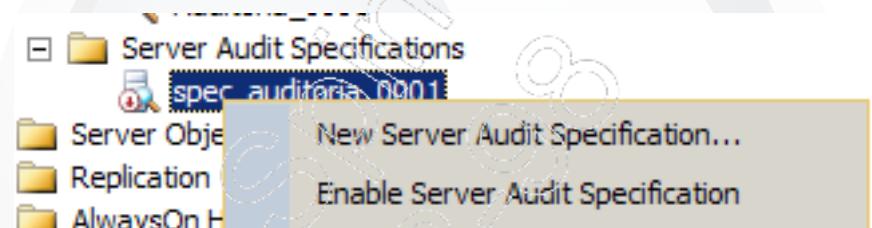
CREATE SERVER AUDIT SPECIFICATION [spec_auditoria_0001]
FOR SERVER AUDIT [Auditoria_0001]
ADD (SCHEMA_OBJECT_CHANGE_GROUP)
WITH (STATE = ON)
GO
```

SQL 2014 - Módulo III

8. Habilite a auditoria, clicando com o botão direito do mouse sobre **Auditoria_0001** e selecionando a opção **Enable Audit**:



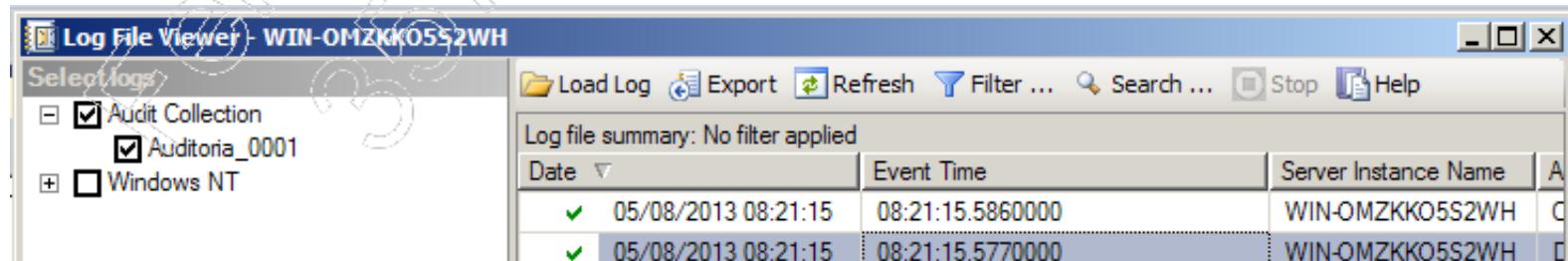
9. Habilite a especificação da auditoria, selecionando a opção **Enable Server Audit Specification...**:



10. Utilize o código a seguir para criar uma tabela chamada **Auditoria** no banco de dados **Pedidos**:

```
CREATE TABLE AUDITORIA (ID      INT PRIMARY KEY, DATA_AUDIT     DA-
TETIME, USUARIO    VARCHAR(50));
```

11. Selecione as opções **Audits** e **Auditoria_0001** e escolha **View Audit Logs**:



12. Verifique os comandos que foram realizados.

Monitorando e ajustando a performance do SQL Server

11

- ✓ Considerações para uma boa performance;
- ✓ Fatores que afetam o tempo de resposta;
- ✓ Planejando o ajuste de performance;
- ✓ Ajustando a performance de uma aplicação;
- ✓ Ferramentas de monitoramento.



IMPACTA
EDITORA

11.1. Introdução

A performance de um sistema é determinada pelo tempo que ele leva para retornar aos usuários as respostas às suas solicitações. O tempo que os usuários esperam para receber tais respostas determina se o sistema é considerado rápido ou lento. É importante que as aplicações apresentem uma boa performance, pois, mesmo que seja bastante útil e prático, um sistema pode não ser aceito pelos usuários pelo fato de apresentar lentidão.

Desde a fase de projeto de um sistema, devemos estar atentos às questões que envolvem a performance a fim de que ele comece a ser executado de forma bem estruturada. No entanto, no decorrer da vida útil de um sistema, ele pode sofrer uma queda de performance devido a fatores como o acesso concorrente de usuários, à quantidade de dados armazenados e manipulados por ele, à fragmentação de dados por conta de constantes atualizações, entre outros. Portanto, o monitoramento e o ajuste da performance são tarefas de manutenção que devem ser realizadas com frequência.

11.2. Considerações para uma boa performance

Durante a fase de projeto de um sistema, alguns fatores são essenciais para determinar sua boa performance. Dentre eles, destacamos a modelagem dos dados e as regras de normalização e de desnormalização. Quando o sistema é projetado levando-se em consideração esses fatores, é bastante provável que ele apresente um bom desempenho no decorrer de sua utilização.

11.3. Fatores que afetam o tempo de resposta

Conforme mencionado anteriormente, o fator mais comum que permite determinar se a performance do sistema está ou não adequada é o tempo que o usuário deve esperar para receber respostas às suas solicitações. Esse tempo de resposta pode variar conforme os seguintes fatores:

- A capacidade física apresentada pelos dispositivos de hardware;
- A competição pelos registros de dados;
- A otimização do banco de dados e da aplicação;
- A quantidade de atividades realizadas pelo servidor;
- A quantidade de queries processadas de forma concorrente.

11.3.1. O que fazer para diminuir o tempo de resposta

Para reduzir o tempo que o usuário espera para obter respostas às solicitações feitas ao sistema, devemos estar atentos à realização de algumas tarefas importantes, a destacar:

- **Reduções**

A performance do sistema pode ser otimizada se conseguirmos reduzir a quantidade de acessos ao disco, o tempo que a CPU leva para executar um determinado processo, o tráfego de dados na rede e a concorrência. Vale destacar que a redução da concorrência requer que sejam escritas transações mais curtas.

- **Hardware**

A redução do tempo de resposta do sistema e a consequente otimização de sua performance são tarefas que requerem o planejamento de alguns itens de forma adequada à aplicação: a velocidade de acesso ao disco, a quantidade de memória disponível e a capacidade apresentada pela CPU.

- **Configurações e gerenciamentos**

Um fator bastante importante para reduzir o tempo de resposta do sistema é determinar uma configuração adequada ao sistema operacional e ao gerenciador de banco de dados. Além disso, também é preciso um gerenciamento contínuo da fragmentação dos dados e uma constante atualização das estatísticas do sistema.

- **Queries**

As queries também exercem um papel importante na redução de tempo de resposta do sistema. Por isso, devemos escrever queries otimizadas, as quais devem ser capazes de obter somente os dados necessários e cujos índices sejam úteis. Devemos ter em mente que a quantidade de queries processadas de forma concorrente e o aumento no tempo de resposta podem ser gerados devido a uma grande quantidade de usuários acessando o sistema. Dessa forma, é preciso estar atento a esse número de queries para que ele possa ser contido caso necessário.

Quando a performance do sistema não se apresenta da forma desejada, podemos ajustá-la. Para isso, é necessário o conhecimento a respeito de alguns fatores importantes, descritos a seguir:

- Quão seletivos são os dados;
- Como a aplicação é utilizada em sistemas OLTP e OLAP;
- Qual a estrutura física e lógica dos dados e como eles são utilizados;
- Qual o ambiente em que a aplicação é executada, bem como seus usuários e dados;
- Quais são as queries executadas pelos usuários e quantas queries são processadas pelo servidor de forma concorrente.

11.4. Planejando o ajuste de performance

O planejamento para ajustar a performance do sistema deve ser realizado com base no volume de atividades executadas pelo servidor, na capacidade física apresentada pelo hardware, na quantidade de queries que deve ser processada, na forma como a aplicação e o banco de dados foram projetados e no fator de contenção.

11.4.1. Situação atual do sistema e objetivos a serem alcançados

Para planejar o ajuste da performance apresentada pelo sistema, devemos definir não apenas seus parâmetros ideais, mas também seus parâmetros atuais. A definição dos parâmetros atuais requer o monitoramento do sistema durante certo período de tempo e, posteriormente, o registro dos resultados obtidos com tal monitoramento.

Esses resultados permitem verificar dados importantes, como os seguintes:

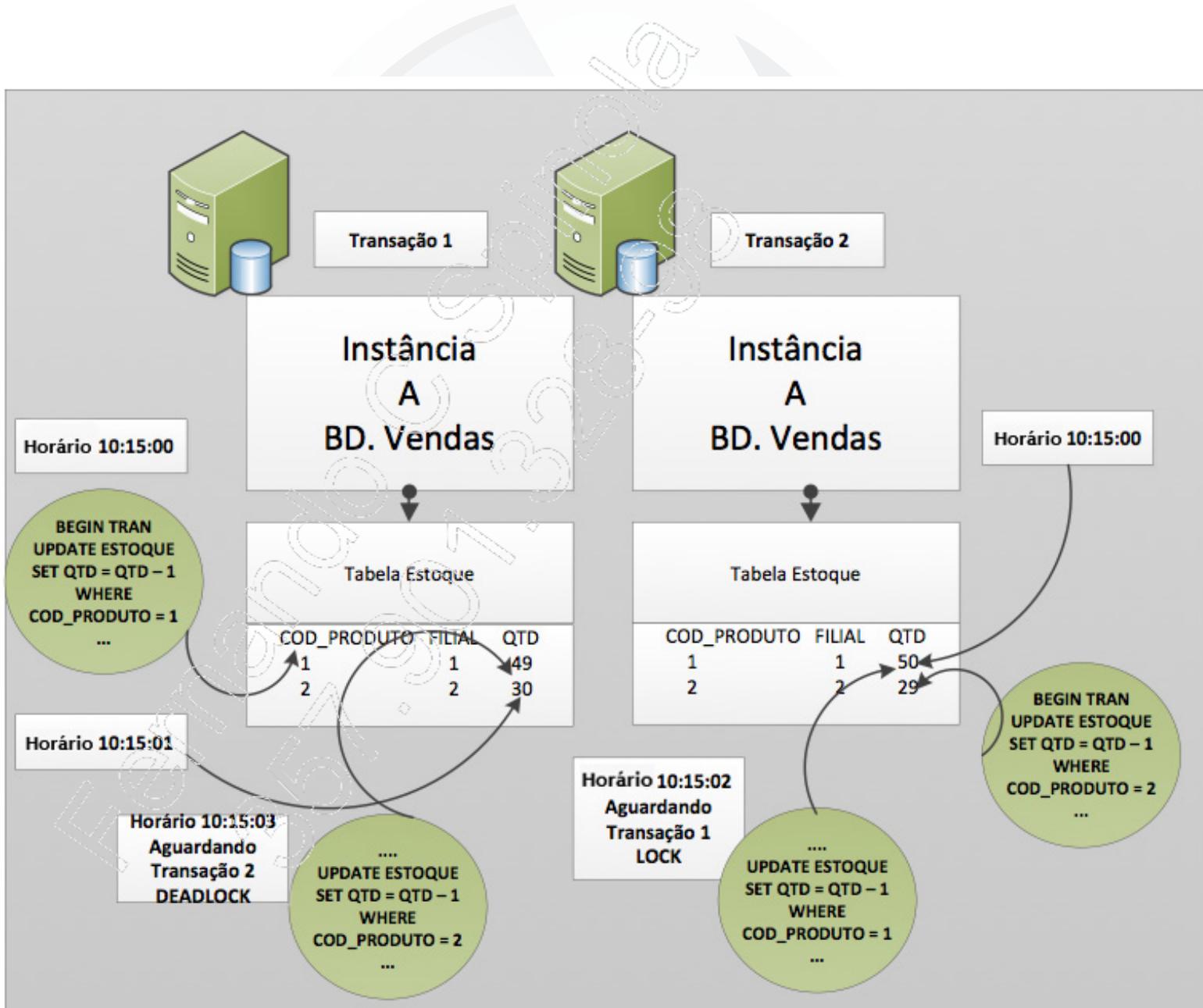
- Quanto tempo durou a execução de tarefas essenciais, como o backup e a restauração;
- Os momentos em que as atividades referentes ao banco de dados atingiram seus picos;
- O tempo de resposta para o processamento de queries e de comandos em batch.

Após conhecer os parâmetros atuais e determinar os ideais, ambos podem ser comparados para que possamos definir quais são os ajustes necessários. Quando encontramos parâmetros ideais bastante diferentes dos parâmetros atuais, estes últimos devem ser testados com maior nível de detalhamento.

11.5. Ajustando a performance de uma aplicação

Para que possamos ajustar a performance de uma aplicação, devemos verificar se os índices existentes são úteis, se somente os dados necessários são obtidos pelas queries, se há ocorrência e contenção de deadlocks e a fragmentação de dados.

A seguir, ilustraremos as ocorrências de locks e deadlocks entre duas transações:



Outros ajustes também necessários são:

- **Ajuste do banco de dados:** O ajuste do banco de dados é uma tarefa que permite otimizar o tempo de resposta das queries. Esse ajuste requer somente o refinamento dos projetos lógico e físico do banco de dados;
- **Ajuste do SQL Server:** Este tipo de ajuste requer a avaliação do projeto de armazenamento e de determinadas opções de configuração;
- **Ajuste da configuração do hardware:** Este tipo de ajuste permite otimizar a performance apresentada pelo sistema. São várias as tarefas que podemos realizar com a finalidade de ajustar a configuração do hardware, por exemplo, utilização de um disco rígido mais rápido, inclusão de mais memória, aumento da performance da rede, entre outras;
- **Identificação de gargalos:** Um gargalo é definido como um componente ou uma atividade que provoca uma limitação da performance de um sistema. Embora qualquer sistema apresente gargalos, eles não podem provocar uma queda de performance que esteja abaixo do ideal. Esta é a importância do monitoramento de tais gargalos;
- **Estratégias de índices e de queries:** Estas estratégias devem determinar que sejam criados índices úteis e que sejam escritas queries capazes de obter somente os dados que são necessários. Isso permite ao SQL Server criar um plano de execução de queries otimizado;
- **Agendamento adequado de tarefas de manutenção:** Estas tarefas devem ser agendadas de forma que sua execução seja frequente e fora do horário de pico da utilização do sistema.

11.6. Ferramentas de monitoramento

Algumas ferramentas são ideais para o monitoramento de determinados itens. A seguir, teremos a relação dos itens a serem monitorados e das ferramentas adequadas para realizar esta tarefa:

- Windows System Monitor e Windows 2000, 2003, 2008, 2012 e XP Event Viewer são as ferramentas ideais para realizar o monitoramento do hardware, do sistema operacional e da aplicação no nível do sistema;
- Job Activity Monitor do SQL Server Management Studio, SQL Profiler, system stored procedures e comandos Transact-SQL são ferramentas ideais para realizar o monitoramento das atividades executadas e da consistência dos dados do SQL Server;
- SQL Profiler, SQL Server Management Studio e Database Engine Tuning Advisor são ferramentas ideais para realizar o monitoramento de índices úteis, de quantidade de entrada/saída e de tempo da CPU com relação às queries.

11.6.1. Windows System Monitor

Esta é uma ferramenta do sistema operacional com a qual podemos realizar o monitoramento da performance e verificar as atividades realizadas no servidor por meio de contadores. Os contadores oferecidos por System Monitor permitem monitorar itens como memória e processador. Embora seja possível acrescentar outros contadores junto àqueles que já existem, temos à disposição vários contadores que permitem monitorar o sistema.

As atividades realizadas pelo SQL Server e os recursos de sistema que ele utiliza também podem ser monitorados por meio de contadores, visto que, durante a instalação do SQL Server, ele adiciona seus contadores à ferramenta System Monitor.

11.6.1.1. Contadores mais relevantes

Alguns contadores de System Monitor são utilizados com mais frequência para monitorar o SQL Server, pois eles fornecem informações que permitem saber o que está acontecendo no servidor. O SQL Server pode utilizar uma quantidade fixa de memória ou pode ajustar essa quantidade de acordo com a memória utilizada por outros aplicativos de forma concorrente com o SQL Server. Para monitorar não apenas essa utilização da memória, mas também o uso de arquivos de paginação, destacamos os seguintes contadores:

- **Contadores de memória**
 - **Available Bytes**: Este contador demonstra a quantidade de memória física, em bytes, que está disponível. Essa quantidade refere-se ao valor mais recente e não a uma média. Quando esse valor é menor que 5000 KB, a quantidade de memória pode ser insuficiente. Portanto, valores maiores são considerados melhores;
 - **Pages/sec**: Este contador tem a finalidade de monitorar a quantidade de acessos à memória virtual (page file) por parte do sistema, isto é, a quantidade de paginação existente no sistema. O valor deste contador deve ser sempre zero, pois, caso contrário, o valor indicará que o servidor está trabalhando com memória insuficiente;
 - **Page Faults/sec**: Este contador tem a função de apresentar valores que registram as falhas de páginas que ocorrem no momento em que um processo necessita de códigos ou de dados que não se encontram em sua memória física. Quando a página que falta encontra-se em outro lugar da memória física, ocorre uma soft fault que, mesmo em grande quantidade, não atrapalha o funcionamento dos processadores. No entanto, quando a falta de uma página requer o acesso ao disco, ocorre uma hard fault, que pode causar um atraso significante quando acontece em grande quantidade, indicando a ocorrência de muita paginação no sistema.

- **Contadores de processos**
 - **Working Set:** Este contador tem a função de monitorar a área da memória recentemente utilizada por um processo. Ele indica a quantidade de memória que está sendo utilizada atualmente e é considerado um contador de grande utilidade aos processos do SQL Server. O valor obtido com este contador deve ser sempre superior a 5000 KB, pois isso indica que as páginas são mantidas no Working Set do processo mesmo que ele não as utilize. Caso contrário, as páginas são retiradas do Working Set. Quando isto acontece, caso essas páginas sejam necessárias, elas serão procuradas em outro lugar da memória física e novamente colocadas no Working Set. Esse processo recebe o nome de faulted back;
 - **Private Bytes:** Este contador realiza o monitoramento da quantidade de bytes que está alocada por um processo e que, sendo assim, estes bytes não podem ser compartilhados com outros processos;
 - **Virtual Bytes:** Este contador realiza o monitoramento da quantidade de memória virtual utilizada por um processo. Quando uma grande quantidade é utilizada, o processo pode não ser capaz de carregar bibliotecas, possibilitando a ocorrência do erro **Out Of Virtual Memory**;
 - **% Processor Time:** Este contador determina a porcentagem de tempo que a CPU é utilizada em um processo.
- **Contadores relacionados a processador**
 - **% Processor Time:** Este contador tem a finalidade de monitorar a utilização total da CPU em nível de sistema. O valor obtido com ele deve ser sempre menor que 90%, embora seja normal a ocorrência de picos que atinjam 100% durante um curto período;

! Este contador de processador possui o mesmo nome que o contador **% Processor Time**, relacionado a processos.

- **% Privileged Time:** Este contador apresenta o tempo que o processador levou para realizar comandos do kernel do sistema operacional. O modo privilegiado é um modo de processamento designado aos componentes do sistema operacional e à manipulação dos drivers do hardware. Ele permite acesso direto ao hardware e à memória. Este contador pode apresentar um valor alto nas situações em que ocorre uma grande quantidade de interrupções geradas por falhas de dispositivos;
- **% User Time:** Este contador tem a função de registrar o tempo que a CPU é utilizada para executar os processos de usuário, como o próprio SQL Server. Ele exibe a média de tempo de utilização da CPU com um determinado processo e dentro de um determinado período.
- **Contadores ligados ao sistema**
 - **Processor Queue Length:** Este contador tem a função de determinar o tamanho da fila de processos. Vale destacar que existe somente uma fila de processos que aguardam pelo processador. Quando esta fila é formada por mais de dois threads, geralmente, isso indica que o processador está congestionado. Este contador exibe somente o último valor observado;
 - **Context Switches/sec:** Este contador determina uma mudança de contexto. Ele monitora a quantidade de vezes que o processador alternou de um thread para outro.
- **Contadores ligados ao disco**
 - **% Disk Time:** Este contador determina o tempo de disco que foi utilizado para os processos de leitura e gravação. O valor obtido com este contador deve ser menor que 90%;
 - **Avg. Disk Queue Length:** Este contador determina a quantidade média de requerimentos de leitura e gravação que são colocados na fila de utilização do disco em um determinado intervalo de tempo;

- **Current Disk Queue Length:** Este contador tem a função de determinar a quantidade de requerimentos de acesso ao disco que estão pendentes no momento. Os picos de acesso ao disco são aceitos somente no momento em que ocorrem determinadas operações, como um checkpoint. No entanto, normalmente, esta fila não deve ser muito grande. Caso haja, constantemente, mais de uma operação na fila, isso pode indicar a sobrecarga do disco;
 - **Disk Reads/sec:** Este contador tem a função de representar a quantidade de operações de leitura, sendo que o valor referente a esta quantidade deve ser menor do que a capacidade física do disco;
 - **Disk Writes/sec:** Este contador tem a função de representar a quantidade de operações de gravação, sendo que o valor referente a essa quantidade deve ser menor do que a capacidade do subsistema de disco rígido.
-
- **Contadores ligados ao SQL Server**
 - **Buffer Manager: Total Pages:** Este contador tem a finalidade de determinar a quantidade total de páginas no cache de buffer. Portanto, nesta contagem são consideradas as páginas utilizadas por outros processos, as páginas de bancos de dados e, até mesmo, as páginas livres;
 - **Buffer Manager: Buffer Cache Hit Ratio:** Este contador tem a finalidade de determinar a quantidade de páginas no cache de buffer, sendo que o valor referente a essa quantidade deve ser maior que 90%;
 - **Buffer Manager: Page Read/sec:** Este contador tem a função de determinar a quantidade de páginas lidas no banco de dados;
 - **Buffer Manager: Page Write/sec:** Este contador tem a função de determinar a quantidade de páginas de banco de dados que são gravadas no disco;
 - **Buffer Manager: Checkpoint Writes/sec:** Este contador tem a função de determinar a quantidade de páginas que estão liberadas para checkpoint ou para qualquer outra operação que necessite de todas as dirty pages;

- **Memory Manager: Total Server Memory:** Este contador tem a função de determinar a quantidade total de memória dinâmica utilizada pelo SQL Server no momento.

11.6.1.2. Pontos de atenção

Destacamos os seguintes pontos de atenção:

- **Contenções**

Há situações que podem representar algum tipo de contenção. Por exemplo, quando o contador de sistema **Processor Queue Length** exibe uma fila com vários processos aguardando pelo processador, quando o contador de processador **% Processor Time** apresenta um valor próximo a 100% ou quando o contador de sistema **Context Switches/sec** apresenta um valor muito alto.

- **Entrada e Saída (I/O)**

Este tipo de operação deve ter sua quantidade reduzida quando os contadores de disco estiverem registrando valores muito altos. Esta redução requer que as seguintes tarefas sejam executadas:

- Criação de índices que sejam úteis;
- Utilização de um disco rígido que apresente mais velocidade;
- Utilização de um servidor adicional com o qual devemos dividir a função de armazenar os dados;
- Transferência de alguns arquivos para outro disco;
- Revisão da modelagem de dados com a aplicação da normalização;
- Ajuste da fragmentação de dados;
- Aumento da capacidade de memória para evitar excesso de paginação;
- Colocação do Transaction Log em um disco diferente daquele em que se encontra o arquivo de dados.

11.6.2. SQL Profiler

Esta ferramenta gráfica permite realizar o monitoramento dos eventos referentes ao SQL Server, como eventos de auditoria, eventos de crescimento ou redução de arquivos pertencentes ao banco de dados, entre outros.

A utilização de **SQL Profiler** requer a especificação dos eventos a serem monitorados e a criação de um Trace com essas informações. Os resultados gerados por esse Trace podem ser salvos em uma tabela de um banco de dados ou em um arquivo para que possam ser verificados posteriormente.

Veja, a seguir, as categorias de eventos com as quais contamos e suas respectivas descrições:

- **Broker:** Esta categoria possui classes de eventos que o Service Broker gera;
- **Cursors:** Esta categoria possui classes de eventos que as operações referentes ao cursor geram;
- **CLR:** Esta categoria possui classes de eventos que a execução de objetos CLR.NET gera;
- **Database:** Esta categoria possui classes de eventos que são geradas de forma automática no momento em que ocorre um aumento ou uma redução dos arquivos de log ou de dados;
- **Deprecation:** Esta categoria possui eventos referentes a objetos e recursos que não devem mais ser utilizados;
- **Error and Warnings:** Esta categoria possui classes de eventos geradas no momento em que um erro ou um alerta do SQL Server ocorre;
- **Full Text:** Esta categoria possui classes geradas no momento em que as buscas do tipo full-text são iniciadas, paralisadas ou simplesmente interrompidas;
- **Locks:** Esta categoria possui classes de eventos geradas no momento em que qualquer tarefa é realizada sobre um lock, bem como no momento em que um lock é adquirido, liberado ou cancelado;

- **Objects:** Esta categoria possui classes de eventos geradas no momento em que criamos, apagamos, abrimos, fechamos ou excluímos um objeto de banco de dados;
- **OLEDB:** Esta categoria possui classes de eventos geradas por uma chamada OLEDB;
- **Performance:** Esta categoria possui classes de eventos geradas no momento da execução de operadores DML;
- **Progress Report:** Esta categoria possui a classe de eventos **Progress Report: Online Index Operation**;
- **Scans:** Esta categoria possui classes de eventos geradas no momento da varredura de tabelas e de índices;
- **Security Audit:** Esta categoria possui classes de eventos utilizadas para auditar as atividades realizadas pelo servidor;
- **Server:** Esta categoria possui diversos eventos referentes ao servidor;
- **Sessions:** Esta categoria possui classes de eventos geradas quando o cliente conecta-se ou desconecta-se de uma instância do SQL Server;
- **Stored Procedures:** Esta categoria possui classes de eventos geradas no momento em que as stored procedures são executadas;
- **Transactions:** Esta categoria possui classes de eventos geradas no momento em que as transações DTC são executadas ou quando elas são gravadas no Transaction Log;
- **TSQL:** Esta categoria possui classes de eventos geradas no momento em que os comandos Transact-SQL passados do cliente para uma instância do SQL Server são executados;
- **User-Configurable:** Esta categoria possui as classes de evento que podem ser definidas pelo usuário.

11.6.2.1. Broker

Vimos que uma das categorias de evento com as quais contamos é a **Broker**. Esta categoria possui diversos eventos **Service Broker**, os quais serão descritos a seguir:

- **Activation**: Este evento é produzido no momento em que o monitor de filas inicia uma stored procedure de ativação;
- **Connection**: Este evento é produzido com a finalidade de fornecer informações a respeito de uma conexão de transporte que o **Service Broker** gerou;
- **Conversation**: Este evento é produzido com a finalidade de fornecer informações a respeito do progresso de uma conversação;
- **Conversation Group**: Este evento é produzido no momento em que cria ou derruba um grupo de conversação;
- **Corrupted Message**: Este evento é produzido com a finalidade de informar que uma mensagem corrompida foi enviada ao banco de dados;
- **Forwarded Message Dropped**: Este evento é produzido no momento em que o SQL Server exclui uma mensagem do **Service Broker** que deveria ter sido enviada;
- **Forwarded Message Sent**: Este evento é produzido no momento em que o SQL Server envia uma mensagem do **Service Broker**;
- **Message Classify**: Este evento é produzido no momento em que o **Service Broker** define o destino da mensagem;
- **Message Drop**: Este evento é produzido no momento em que o **Service Broker** não pode manter uma mensagem que ele recebeu, mas que deveria ter sido enviada a um serviço na mesma instância;
- **Remote Message Ack**: Este evento é produzido no momento em que o **Service Broker** recebe a confirmação de uma mensagem.

11.6.2.2. Cursor

Esta categoria possui as seguintes classes de evento:

- **CursorClose**: Esta classe é responsável por criar um registro no Trace nas situações em que um cursor presente no comando Transact-SQL é fechado pelo ODBC, pelo OLE DB ou pelo DB-Library;
- **CursorExecute**: Esta classe é responsável por criar um registro no Trace quando um cursor presente no comando Transact-SQL é executado pelo ODBC, pelo OLE DB ou pelo DB-Library;
- **CursorImplicitConversion**: Esta classe tem a função de criar um registro no Trace nas situações em que um cursor presente no comando Transact-SQL é convertido de um tipo para outro pelo SQL Server;
- **CursorOpen**: Esta classe tem a finalidade de criar um registro no Trace quando um cursor presente no comando Transact-SQL é aberto pelo ODBC, pelo OLE DB ou pelo DB-Library;
- **CursorPrepare**: Esta classe tem a função de criar um registro no Trace nas situações em que um cursor presente no comando Transact-SQL é preparado pelo ODBC, pelo OLE DB ou pelo DB-Library;
- **CursorRecompile**: Esta classe tem a função de criar um registro no Trace quando um cursor aberto no comando Transact-SQL pelo ODBC ou pelo DB-Library é recompilado por conta de alterações feitas no esquema. Essa recompilação pode ser feita de forma direta ou indireta;
- **CursorUnprepare**: Esta classe tem a função de criar um registro no Trace nas situações em que um cursor presente no comando Transact-SQL é excluído pelo ODBC, pelo OLE DB ou pelo DB-Library.

11.6.2.3. CLR

Esta categoria possui a classe de evento **Assembly Load**, a qual indica que a solicitação para carregar um assembly foi executada. Portanto, esta classe é criada no momento dessa execução.

11.6.2.4. Database

Esta categoria possui as seguintes classes de evento:

- **Data File Auto Grow:** Esta classe tem a função de criar um registro no arquivo de Trace quando o crescimento de arquivos de um banco de dados ocorre de forma automática;
- **Data File Auto Shrink:** Esta classe tem a função de criar um registro no Trace nas situações em que a redução de arquivos de um banco de dados ocorre de forma automática;
- **Database Mirroring State Change:** Esta classe tem a finalidade de indicar o momento em que há uma alteração no estado de um banco de dados espelhado;
- **Log File Auto Grow:** Esta classe tem a função de criar um registro no Trace nas situações em que os arquivos de log de um banco de dados crescem de forma automática;
- **Log File Auto Shrink:** Esta classe tem a finalidade de criar um registro no Trace nas situações em que os arquivos de log de um banco de dados são reduzidos de forma automática.

11.6.2.5. Deprecation

Esta categoria possui as seguintes classes de evento:

- **Deprecation Announcement:** Esta classe tem a função de indicar a exclusão de um determinado recurso de uma versão posterior do SQL Server;
- **Deprecation Final Support:** Esta classe tem a finalidade de indicar a exclusão de um determinado recurso do próximo lançamento do SQL Server.

11.6.2.6. Errors and Warnings

Esta categoria possui as seguintes classes de evento:

- **Attention:** Esta classe tem a função de registrar uma linha no Trace quando ocorre um evento que merece atenção, como a interrupção de uma conexão de cliente;
- **Background Job Error:** Esta classe tem a função de indicar que um trabalho executado em segundo plano foi encerrado de forma inadequada;
- **Blocked Process Report:** Esta classe tem a função de indicar que uma tarefa permaneceu bloqueada por mais tempo do que o determinado;
- **ErrorLog:** Esta classe tem a função de registrar uma linha no Trace nas situações em que é registrado um erro do SQL Server em seu **Error Log**;
- **EventLog:** Esta classe tem a finalidade de registrar uma linha no Trace quando é registrado um erro do SQL Server no **Application Log do Event Viewer** do Windows;
- **Exception:** Esta classe tem a finalidade de registrar uma linha no Trace quando ocorre uma exceção no SQL Server;
- **Exchange Spill:** Esta classe tem a finalidade de indicar a gravação em um banco de dados TempDB de buffers de comunicação presentes em um planejamento de query paralelo;
- **Execution Warnings:** Esta classe tem a finalidade de indicar, no decorrer da execução de um comando SQL Server ou de uma stored procedure, a ocorrência de advertências da memória;
- **Hash Warnings:** Esta classe tem a função de registrar uma linha no Trace nas situações em que as operações hash apresentam algum tipo de problema;
- **Missing Column Statistics:** Esta classe tem a função de registrar uma linha no Trace quando o otimizador de queries é incapaz de encontrar as estatísticas de colunas;

- **Missing Join Predicate:** Esta classe tem a função de registrar uma linha no Trace nas situações em que a execução da query com **Join** apresenta lentidão devido à falta de um predicado;
- **User Error Message:** Esta classe tem a função de exibir as mensagens de erro que o usuário visualiza;
- **Sort Warnings:** Esta classe tem a função de registrar uma linha no Trace quando são executadas operações de ordenação de dados que não permanecem fixas na memória.

11.6.2.7. Full Text

Esta categoria possui as seguintes classes de evento:

- **Crawl Aborted:** Esta classe tem a finalidade de indicar que o crawl full-text foi abortado devido ao fato de ter encontrado uma exceção;
- **Crawl Started:** Esta classe indica o início de um crawl full-text;
- **Crawl Stopped:** Esta classe indica a finalização de um crawl full-text.

11.6.2.8. Locks

Esta categoria possui as seguintes classes de evento:

- **Deadlock: Graph:** Esta classe tem a finalidade de fornecer uma descrição XML referente a um deadlock;
- **Lock:Acquired:** Esta classe tem a função de registrar uma linha no Trace nas situações em que um lock é adquirido em um dos recursos do SQL Server;
- **Lock:Cancel:** Esta classe tem a função de registrar uma linha no Trace nas situações em que ocorre o cancelamento do lock em um dos recursos do SQL Server;
- **Lock:Deadlock:** Esta classe tem a finalidade de registrar uma linha no Trace quando ocorre um deadlock no SQL Server;

- **Lock:Deadlock Chain:** Esta classe tem a função de registrar uma linha no Trace nas situações em que ocorrem deadlocks;
- **Lock:Escalation:** Esta classe tem a finalidade de registrar uma linha no Trace quando um tipo de lock é convertido para outro de forma automática;
- **Lock:Released:** Esta classe tem a função de registrar uma linha no Trace nas situações em que um lock presente em um recurso é liberado;
- **Lock:Timeout:** Esta classe tem a finalidade de registrar uma linha no Trace nas situações em que ocorre um timeout (intervalo) devido à espera pela liberação de recursos que possuem locks;
- **Lock:Timeout (timeout > 0):** Esta classe tem a função de registrar uma linha no Trace quando as solicitações por locks não são completadas devido ao fato de o recurso solicitado estar bloqueado por outra transação. Isso ocorre somente nas situações em que o valor de timeout do lock é maior que zero.

11.6.2.9. Objects

Esta categoria possui as seguintes classes de evento:

- **AutoStats:** Esta classe tem a finalidade de registrar uma linha no Trace nas situações em que o SQL Server cria ou altera as estatísticas de forma automática;
- **Object:Closed:** Esta classe tem a função de registrar uma linha no Trace quando um objeto no SQL Server é fechado;
- **Object:Created:** Esta classe tem a função de registrar uma linha no Trace nas situações em que um objeto é criado no SQL Server;
- **Object:Deleted:** Esta classe tem a finalidade de registrar uma linha no Trace quando um objeto é excluído do SQL Server;
- **Object:Altered:** Esta classe tem a função de indicar que uma alteração foi realizada sobre o objeto.

11.6.2.10. OLE DB

Esta categoria possui as seguintes classes de evento:

- **OLEDB Call:** Esta classe tem a função de indicar que o SQL Server fez uma chamada a um provedor OLE DB em busca de stored procedures remotas e queries distribuídas. É possível utilizar essa classe no Trace quando se deseja monitorar chamadas que não são feitas ao método **QueryInterface** ou que não solicitam dados;
- **OLEDB DataRead:** Esta classe tem a função de indicar que o SQL Server fez uma chamada a um provedor OLE DB em busca de stored procedures remotas e queries distribuídas;
- **OLEDB Errors:** Esta classe tem a finalidade de indicar que um erro foi retornado a partir de uma chamada feita a um provedor OLE DB;
- **OLEDB Provider Information:** Esta classe tem a finalidade de indicar que uma query distribuída foi executada e coletou dados correspondentes à conexão com o provedor;
- **OLEDB QueryInterface:** Esta classe tem a finalidade de indicar que o SQL Server fez uma chamada ao **QueryInterface** OLE DB por stored procedures remotas e queries distribuídas.

11.6.2.11. Performance

Esta categoria possui as seguintes classes de evento:

- **Degree of Parallelism (7.0 Insert):** Esta classe tem a finalidade de indicar que um comando INSERT foi executado pelo SQL Server;
- **Performance Statistics:** Esta classe tem a finalidade de monitorar o desempenho apresentado pelas queries que estão em execução;

- **Showplan All:** Esta classe tem a função de identificar a presença de operadores **Showplan** em um comando SQL;
- **Showplan All for Query Compile:** Esta classe tem a função de exibir dados em tempo de compilação aos operadores **Showplan**;
- **Showplan Statistics Profile:** Esta classe tem a função de exibir o provável custo de uma query;
- **Showplan Text:** Esta classe tem a função de identificar os operadores **Showplan** utilizados nas versões 2000 e 7.0 do SQL Server. As informações obtidas com esta classe são formatadas como dados binários;
- **Showplan Text (Unencoded):** Esta classe tem a mesma função descrita na classe anterior, porém, suas informações são formatadas como strings;
- **Showplan XML:** Esta classe tem a função de identificar a presença de operadores **Showplan** em um comando SQL. Vale destacar que os eventos obtidos com esta classe são armazenados como um documento XML bem estruturado;
- **Showplan XML For Query Compile:** Esta classe tem a função de exibir dados para os operadores **Showplan** em tempo de compilação no formato XML;
- **Showplan XML Statistics Profile:** Esta classe tem a função de identificar os operadores **Showplan** que estão associados a um comando SQL. As informações obtidas como resultado são formatadas como um documento XML;
- **SQL:FullTextQuery:** Esta classe tem a função de indicar que uma query full-text foi executada pelo SQL Server.

11.6.2.12. Security Audit

Esta categoria possui as seguintes classes de evento:

- **Audit Add DB User Event:** Esta classe tem a função de registrar uma linha no Trace quando um usuário é incluído ou excluído de um banco de dados;
- **Audit Add Login to Server Role Event:** Esta classe tem a função de registrar uma linha no Trace nas situações em que um usuário é incluído ou excluído de um role no servidor;
- **Audit Add Member to DB Role Event:** Esta classe tem a função de registrar uma linha no Trace quando um usuário é incluído ou excluído de um role de banco de dados, seja ele fixo ou definido pelo usuário;
- **Audit Add Role Event:** Esta classe tem a função de registrar uma linha no Trace quando um role definido pelo usuário é incluído ou excluído de um banco de dados;
- **Audit AddLogin Event:** Esta classe tem a função de registrar uma linha no Trace nas situações em que um login é incluído ou excluído do SQL Server;
- **Audit App Role Change Password Event:** Esta classe tem a função de registrar uma linha no Trace quando a senha de um Application Role sofre alterações;
- **Audit Backup/Restore Event:** Esta classe tem a função de registrar uma linha no Trace nas situações em que é realizada uma operação de backup ou de restauração de um banco de dados;
- **Audit Broker Conversation:** Esta classe tem a função de enviar mensagens de auditoria referentes à segurança do diálogo de service Broker;
- **Audit Broker Login:** Esta classe tem a função de enviar mensagens de auditoria referentes à segurança do transporte de Service Broker;
- **Audit Change Audit Event:** Esta classe tem a função de registrar alterações ocorridas com a auditoria;

- **Audit Change Database Owner:** Esta classe tem a finalidade de indicar que já foram verificadas as permissões para alterar o schema do banco de dados;
- **Audit Database Management:** Esta classe tem a finalidade de indicar que um banco de dados foi criado, alterado ou excluído;
- **Audit Database Mirroring Login:** Esta classe tem a finalidade de enviar mensagens de auditoria referentes à segurança do transporte de espelhamento de banco de dados;
- **Audit Database Object Access:** Esta classe tem a finalidade de indicar que um objeto de banco de dados foi acessado;
- **Audit Database Object GDR:** Esta classe tem a finalidade de indicar a ocorrência de um evento GDR para um objeto de banco de dados;
- **Audit Database Object Management:** Esta classe tem a finalidade de indicar a execução de um comando CREATE, ALTER ou DROP no objeto de banco de dados;
- **Audit Database Object Take Ownership:** Esta classe tem a finalidade de indicar uma alteração do schema de objetos no escopo do banco de dados;
- **Audit Database Operation:** Esta classe tem a finalidade de indicar a ocorrência de diversas operações, tais como um checkpoint;
- **Audit Database Principal Impersonation:** Esta classe tem a finalidade de indicar a ocorrência de uma personificação dentro do escopo do banco de dados;
- **Audit Database Principal:** Esta classe tem a finalidade de indicar a criação, a alteração ou a exclusão de principais presentes em um banco de dados;
- **Audit Database Scope GDR:** Esta classe tem a finalidade de indicar a emissão de um GRANT, um DENY ou um REVOKE por um usuário no SQL Server a fim de que ele obtenha permissão para um comando;

- **Audit DBCC Event:** Esta classe tem a função de registrar uma linha no Trace quando ocorre a execução de um comando DBCC;
- **Audit Login Event:** Esta classe tem a função de registrar uma linha no Trace nas situações em que ocorre uma conexão no sistema;
- **Audit Login Change Password Event:** Esta classe tem a função de registrar uma linha no Trace quando a senha é alterada no sistema. Com isso, a senha não é registrada;
- **Audit Login Change Property Event:** Esta classe tem a função de registrar uma linha no Trace nas situações em que as propriedades de um login são alteradas, exceto quando a alteração é realizada sobre a senha;
- **Audit Login Failed Event:** Esta classe tem a função de registrar uma linha no Trace nas situações em que um usuário tenta efetuar sua conexão ao sistema utilizando um login inválido;
- **Audit Login GDR Event:** Esta classe tem a função de registrar uma linha no Trace nas situações em que é efetuado um GRANT, um DENY ou um REVOKE sobre uma `sp_grantlogin`, uma `sp_denylogin` ou uma `sp_revokelogin` para os logins que são mapeados do Windows para o SQL Server;
- **Audit Logout Event:** Esta classe tem a função de registrar uma linha no Trace quando um logout é efetuado no sistema;
- **Audit Object Derived Permission Event:** Esta classe tem a função de registrar uma linha no Trace quando um comando CREATE, ALTER ou DROP é executado no sistema;
- **Audit Server Starts and Stops:** Esta classe tem a função de registrar uma linha no Trace nas situações em que os serviços do SQL Server são inicializados, encerrados ou paralisados;
- **Audit Statement Permission Event:** Esta classe tem a função de registrar uma linha no Trace nas situações em que as permissões de comandos são utilizadas;

- **Audit Schema Object Access:** Esta classe tem a finalidade de indicar a utilização das permissões de objetos;
- **Audit Schema Object GDR:** Esta classe tem a finalidade de indicar que um usuário executou um GRANT, um REVOKE ou um DENY para uma permissão de objeto de esquema no SQL Server;
- **Audit Schema Object Management:** Esta classe tem a finalidade de indicar a criação, a alteração ou a exclusão de um objeto de servidor;
- **Audit Schema Object Take Ownership:** Esta classe tem a finalidade de indicar a verificação das permissões para se tornar o proprietário dos objetos de esquema;
- **Audit Server Alter Trace:** Esta classe tem a finalidade de indicar que a verificação da permissão ALTER TRACE foi realizada;
- **Audit Server Object Management:** Esta classe tem a finalidade de indicar que um dos seguintes eventos ocorreu para um objeto de servidor: CREATE, ALTER ou DROP;
- **Audit Server Object Take Ownership:** Esta classe tem a finalidade de indicar que houve alteração quanto ao schema do objeto de servidor;
- **Audit Server Operation:** Esta classe tem a finalidade de indicar que foram realizadas operações Audit no servidor;
- **Audit Server Principal Impersonation:** Esta classe tem a finalidade de indicar que dentro do escopo do servidor foi realizada uma personificação;
- **Audit Server Principal Management:** Esta classe tem a finalidade de indicar que um dos seguintes comandos foi executado para um principal de servidor: CREATE, ALTER ou DROP;
- **Audit Server Scope GDR:** Esta classe tem a finalidade de indicar que ocorreu um evento GDR para as permissões de servidor.

11.6.2.13. Stored Procedures

Esta categoria possui as seguintes classes de evento:

- **RPC:Output Parameter**: Esta classe tem a função de registrar uma linha no Trace que exiba dados a respeito dos parâmetros de saída de uma chamada de stored procedure remota, também conhecida como RPC, que foi executada anteriormente;
- **RPC:Complete**: Esta classe tem a função de registrar uma linha no Trace nas situações em que uma stored procedure remota é completada;
- **RPC:Starting**: Esta classe tem a função de registrar uma linha no Trace quando uma stored procedure remota é iniciada;
- **RPC:CacheHit**: Esta classe tem a função de registrar uma linha no Trace nas situações em que uma stored procedure presente no cache deve ser executada. Vale destacar que a stored procedure **SP:CacheHit** substituiu a **SP:ExecContextHit**, a qual era utilizada até a versão 2000 do SQL Server com a mesma finalidade daquela stored procedure;
- **SP:CacheInsert**: Esta classe tem a função de registrar uma linha no Trace nas situações em que ocorre uma inclusão na procedure cache;
- **SP:CacheMiss**: Esta classe tem a função de registrar uma linha no Trace nas situações em que se deve executar uma stored procedure que não está presente na procedure cache;
- **SP:CacheRemove**: Esta classe tem a função de registrar uma linha no Trace quando um item da procedure cache é excluído;
- **SP:Completed**: Esta classe tem a função de registrar uma linha no Trace nas situações em que ocorre o término da execução de uma stored procedure;
- **SP:Recompile**: Esta classe tem a função de registrar uma linha no Trace quando ocorre a recompilação de uma stored procedure a ser executada;

- **ExistingConnection:** Esta classe tem a finalidade de indicar as propriedades referentes às conexões de usuários existentes no momento em que o Trace tem início;
- **SP:Starting:** Esta classe tem a finalidade de registrar uma linha no Trace quando tem início uma stored procedure a ser executada;
- **SP:StmtCompleted:** Esta classe tem a finalidade de registrar uma linha no Trace nas situações em que é finalizado um comando de dentro de uma stored procedure;
- **SP:StmtStarting:** Esta classe tem a finalidade de registrar uma linha no Trace quando é iniciado um comando de dentro de uma stored procedure.

11.6.2.14. Transactions

Esta categoria possui as seguintes classes de evento:

- **DTCTransaction:** Esta classe tem a função de registrar uma linha no Trace do SQL Profiler nas situações em que as transações distribuídas ocorrem entre dois ou mais bancos de dados;
- **SQLTransaction:** Esta classe tem a função de registrar uma linha no Trace quando os seguintes comandos são executados: BEGIN, COMMIT, SAVE e ROLLBACK TRANSACTION;
- **Transaction Log:** Esta classe tem a função de registrar uma linha no Trace quando ocorrem registros das transações no Transaction Log;
- **TM:Begin Tran Completed:** Esta classe tem a função de indicar que uma solicitação BEGIN TRANSACTION foi finalizada;
- **TM:Begin Tran Starting:** Esta classe tem a função de indicar o início de uma solicitação BEGIN TRANSACTION;
- **TM:Commit Tran Completed:** Esta classe tem a função de indicar que uma solicitação COMMIT TRANSACTION foi finalizada;

- **TM:Commit Tran Starting:** Esta classe tem a finalidade de indicar o início de uma solicitação COMMIT TRANSACTION;
- **TM:Promote Tran Completed:** Esta classe tem a função de indicar que uma solicitação PROMOTE TRANSACTION foi finalizada;
- **TM:Promote Tran Starting:** Esta classe tem a finalidade de indicar o início de uma solicitação PROMOTE TRANSACTION;
- **TM:Rollback Tran Completed:** Esta classe tem a função de indicar que uma solicitação ROLLBACK TRANSACTION foi finalizada;
- **TM:Rollback Tran Starting:** Esta classe tem a finalidade de indicar o início de uma solicitação ROLLBACK TRANSACTION;
- **TM:Save Tran Completed:** Esta classe tem a função de indicar que uma solicitação SAVE TRANSACTION foi finalizada;
- **TM:Save Tran Starting:** Esta classe tem a finalidade de indicar o início de uma solicitação SAVE TRANSACTION.

11.6.2.15. TSQL

Esta categoria possui as seguintes classes de evento:

- **Exec Prepared SQL:** Esta classe tem a função de registrar uma linha no Trace nas situações em que um comando SQL é preparado ou executado pelo ODBC, pelo OLE DB ou pelo DB-Library;
- **Prepare SQL:** Esta classe tem a função de registrar uma linha no Trace nas situações em que um ou mais comandos SQL são preparados para serem utilizados por ODBC, OLE DB ou DB-Library;
- **SQL:BatchStarting:** Esta classe tem a função de registrar uma linha no Trace nas situações em que se inicia um batch de comandos Transact-SQL;

- **SQL:BatchCompleted:** Esta classe tem a função de registrar uma linha no Trace quando se completa um batch de comandos Transact-SQL;
- **SQL:StmtStarting:** Esta classe tem a função de registrar uma linha no Trace nas situações em que se inicia um comando Transact-SQL;
- **SQL:StmtCompleted:** Esta classe tem a função de registrar uma linha no Trace quando se completa um comando Transact-SQL;
- **SQL: StmtRecompile:** Esta classe tem a finalidade de indicar a ocorrência de recompilações no nível do comando, as quais são causadas por todos os tipos de batches.

11.6.3. Transact SQL

Conheceremos, a seguir, os procedimentos e comandos que auxiliam a monitoração do SQL Server.

11.6.3.1. Procedimentos (Stored Procedures)

As seguintes stored procedures podem ser usadas para auxiliar a monitoração do SQL Server:

- **SP_SpaceUsed:** Esta system stored procedure tem a função de retornar o valor referente à quantidade de espaço em disco utilizada por tabelas ou por bancos de dados;
- **SP_HelpDB:** Esta system stored procedure tem a função de retornar não apenas os bancos de dados, mas também os objetos que eles possuem;
- **SP_Monitor:** Esta system stored procedure tem a função de retornar as estatísticas do SQL Server;

- **SP_WHO**: Esta system stored procedure tem a função de retornar os usuários e os processos atuais do SQL Server;
- **SP_LOCK**: Esta system stored procedure tem a função de retornar informações referentes aos impedimentos, aos locks e aos deadlocks;
- **SP_STATISTICS**: Esta system stored procedure tem a função de retornar todos os índices de uma determinada tabela;
- **SP_HELPINDEX**: Esta system stored procedure tem a função de retornar os índices de uma tabela.

11.6.3.2. DBCC

Os seguintes comandos podem ser usados para auxiliar a monitoração do SQL Server:

- **DBCC TRACEON**: Este comando tem a função de habilitar as flags de rastreamento de forma global;
- **DBCC TRACEOFF**: Este comando tem a função de desabilitar as flags de rastreamento que foram configuradas de forma global;
- **DBCC TRACESTATUS**: Este comando tem a função de exibir as flags de rastreamento que foram configuradas de forma global.

Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo.

- A performance de um sistema é determinada pelo tempo que ele leva para retornar aos usuários respostas às suas solicitações. O tempo que os usuários esperam para receber tais respostas determina se o sistema é considerado rápido ou lento;
- Durante a fase de projeto de um sistema, alguns fatores são essenciais para determinar sua boa performance. Dentre eles, destacamos a modelagem dos dados e as regras de normalização e de desnormalização. Quando o sistema é projetado levando-se em consideração esses fatores, é bastante provável que o sistema apresente um bom desempenho no decorrer de sua utilização;
- O tempo de resposta das solicitações feitas ao sistema pode variar conforme os seguintes fatores: a capacidade física apresentada pelos dispositivos de hardware; a competição pelos registros de dados; a otimização do banco de dados e da aplicação; a quantidade de atividades realizadas pelo servidor; a quantidade de queries processadas de forma concorrente;
- O planejamento para ajustar a performance do sistema deve ser realizado com base no volume de atividades executadas pelo servidor, na capacidade física apresentada pelo hardware, na quantidade de queries que devem ser processadas, na forma como foram projetados o banco de dados e a aplicação, e no fator de contenção;
- As ferramentas de monitoramento são facilmente configuradas. Dentre elas, destacamos: System Monitor; SQL Profiler; Activity Monitor; Event Viewer e ferramentas Transact-SQL.

Monitorando e ajustando a performance do SQL Server

11

Teste seus conhecimentos

Fernando
357.90



IMPACTA
EDITORA

1. Quais os fatores que devem ser considerados no planejamento do ajuste da performance do sistema?

- a) O SQL realiza o ajuste de performance de forma automática.
- b) Volume de atividades, projeto e local de armazenamento.
- c) Volume de atividades, hardware, queries, e local de armazenamento.
- d) Hardware, queries, projeto e local de armazenamento.
- e) Volume de atividades, hardware, queries, projeto e local de armazenamento.

2. Qual das alternativas a seguir não é uma ferramenta para análise de performance?

- a) PERFORM
- b) A view SYSOBJECTS.
- c) As stored procedures SP_SPACEUSED, SP_LOCK, SP_MONITOR, SP_WHO, etc.
- d) Job Activity Monitor
- e) Profiler

3. Quais componentes em um servidor são os principais responsáveis por problemas de performance?

- a) Discos
- b) Processadores
- c) Discos, Processadores e Memória.
- d) Memória
- e) Discos e Memória.

4. Com relação à construção das queries afetar a performance, qual das alternativas a seguir não é um dos fatores que devemos considerar?

- a) Utilização em sistemas OLTP e OLAP.
- b) Seletividade dos dados.
- c) Devemos utilizar o modelo OLTP e OLAP no mesmo servidor.
- d) Quais são as queries executadas pelos usuários.
- e) Ambiente da aplicação.

5. Considere a seguinte pergunta: o SQL Profiler permite recuperar informações da instância SQL? Qual alternativa a responde corretamente?

- a) Sim, mas somente do banco corrente.
- b) Não, mas é ótimo como ferramenta de análise.
- c) Sim, mas somente consultas.
- d) Não, somente de consultas.
- e) Sim, o SQL Profiler permite recuperar várias informações a respeito da instância SQL.

Monitorando e ajustando a performance do SQL Server

11

Mãos à obra!

Fernando Spinola
357.907-8



IMPACTA
EDITORA

Laboratório 1

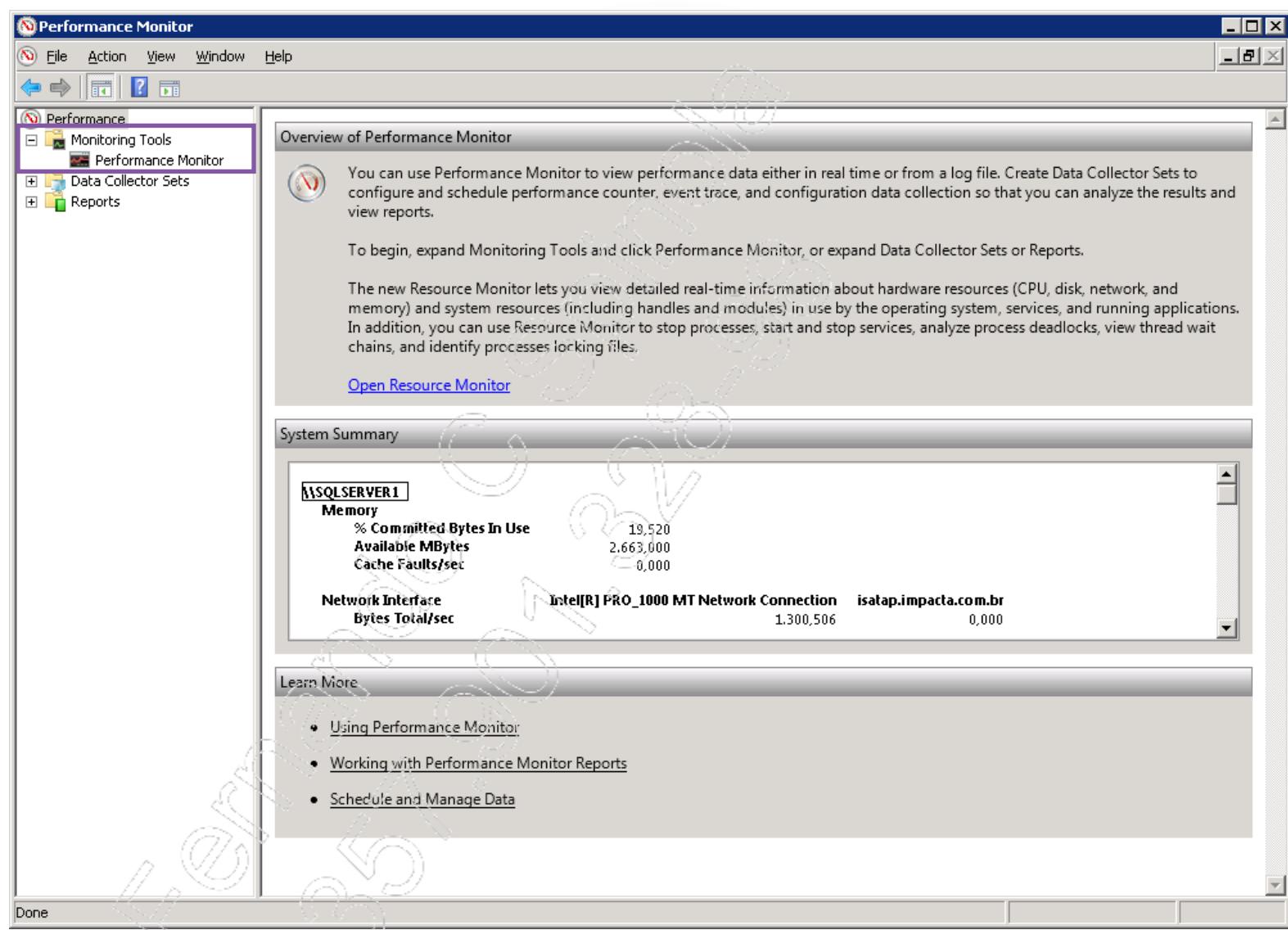
Neste exercício, vamos criar algumas tabelas e inserir nelas alguns dados para termos um ambiente para os nossos testes. Em seguida, vamos utilizar o **Microsoft Performance Monitor** para monitorar as atividades do sistema através de alguns contadores.

A – Preparando o ambiente

1. Abra o **SQL Server Management Studio**. Para tanto, clique no botão **Start**, em seguida, escolha a opção **All Programs**, depois escolha **Microsoft SQL Server 2014** e selecione a opção **SQL Server Management Studio**;
2. Conecte-se com a autenticação do Windows à primeira instância do SQL Server 2014;
3. Na tela que será exibida, na barra de ferramentas, clique na opção **File** e, em seguida, escolha a opção **Open**. Depois, clique em **File** novamente. Abra o **Script_01** da pasta **Capitulo_11**. Conecte-se com a autenticação do Windows à primeira instância do SQL Server 2014;
4. Com o script aberto, pressione F5 para executá-lo;
5. Minimize o SQL Server Management Studio.

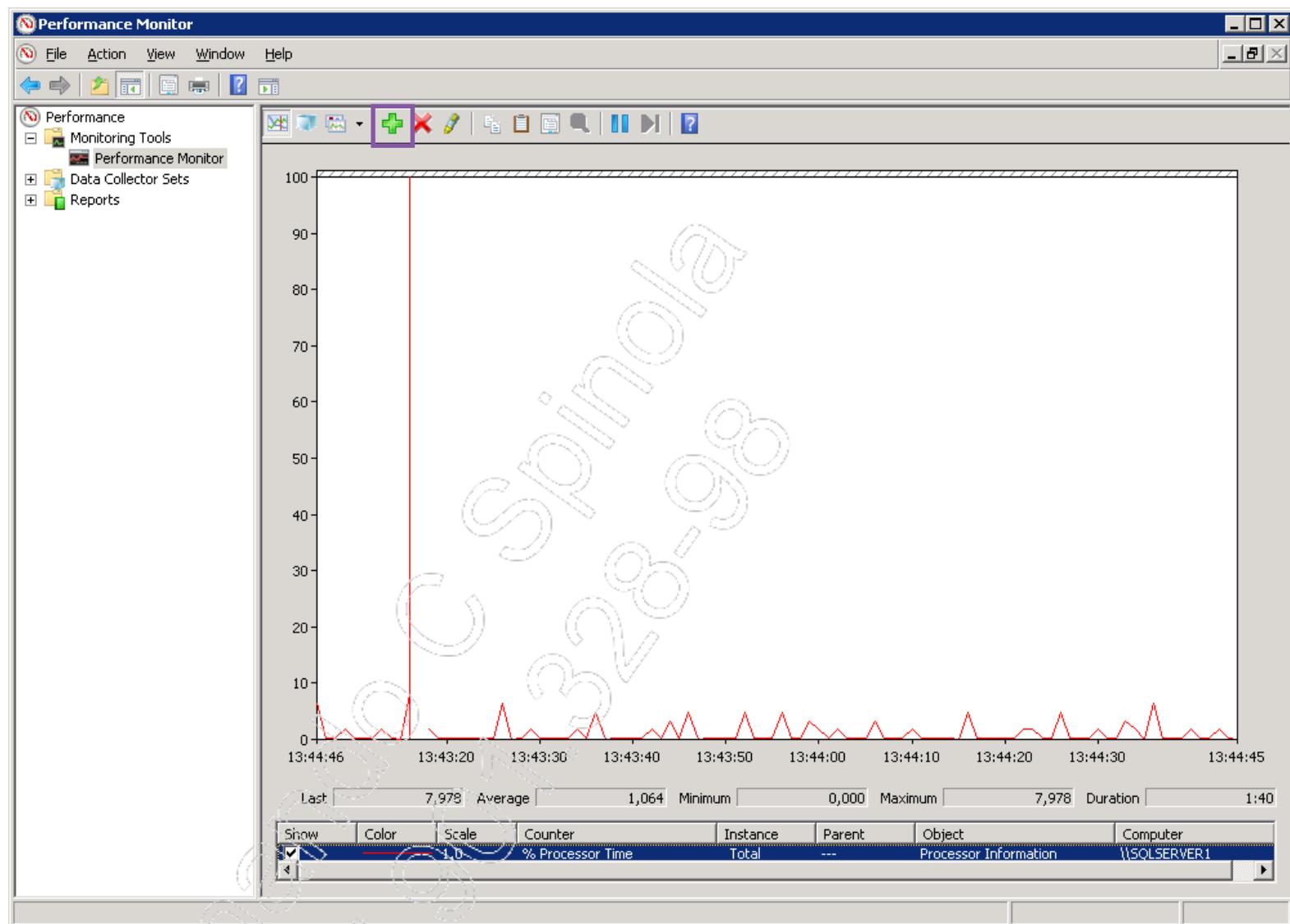
B – Monitorando com o Performance Monitor do Windows

1. Abra o **Performance Monitor**. Para tanto, clique no botão **Start** do Windows, escolha as opções **All Programs**, **Administrative Tools** e **Performance Monitor**;
2. Do lado direito da janela do **Performance Monitor**, na barra de ferramentas, expanda a pasta **Monitoring Tools** e clique em **Performance Monitor**:

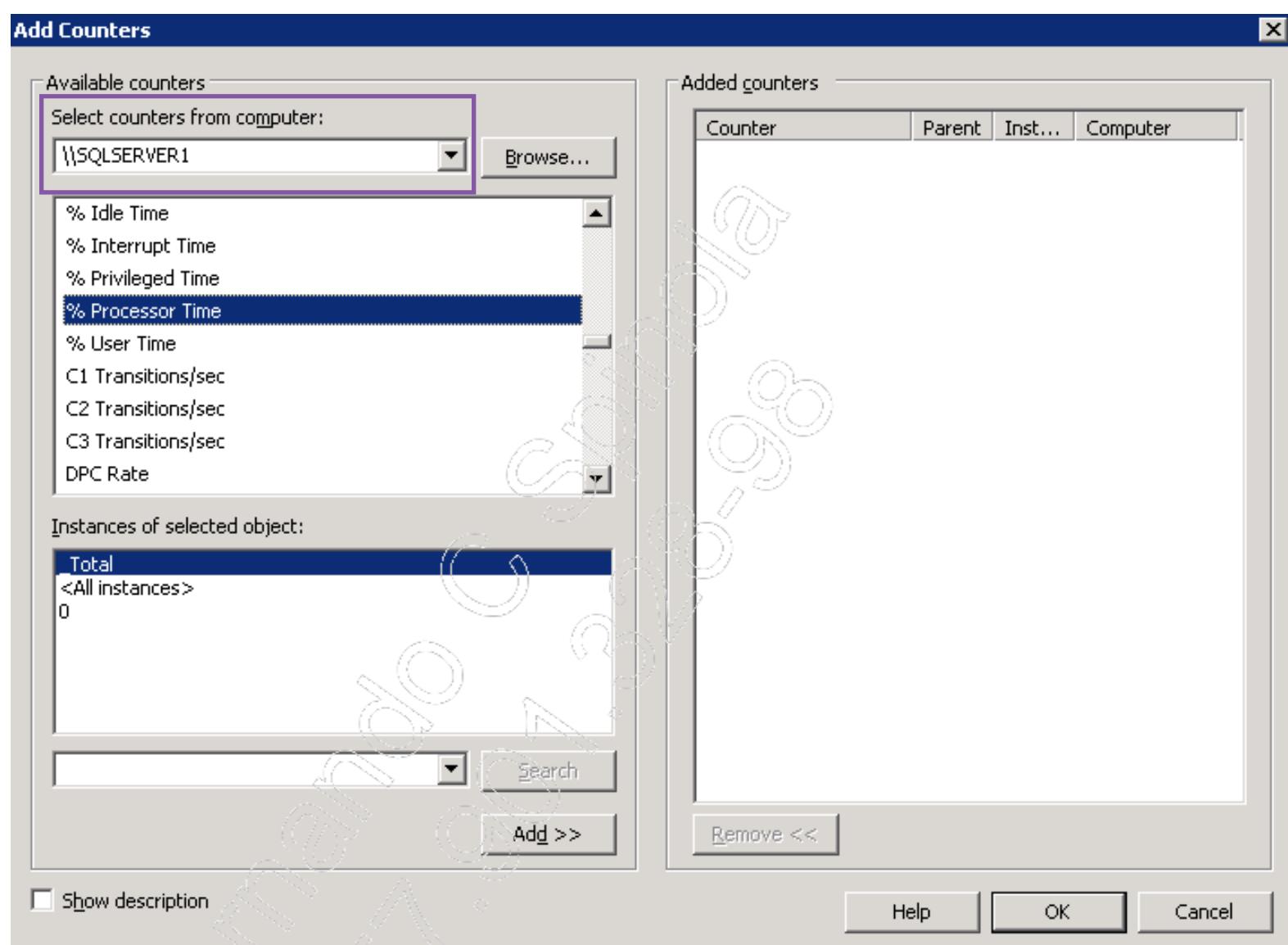


SQL 2014 - Módulo III

3. Ao mudar o conteúdo da janela, observe que aparece uma barra com ferramentas, então, clique no sinal de adição (+) para adicionar contadores a serem monitorados:



4. Na tela seguinte, na opção **Select counters from computer**, deixe selecionado o nome do servidor atualmente usado. Em seguida, selecione alguns itens da listagem de objetos de performance, juntamente com alguns de seus respectivos contadores:

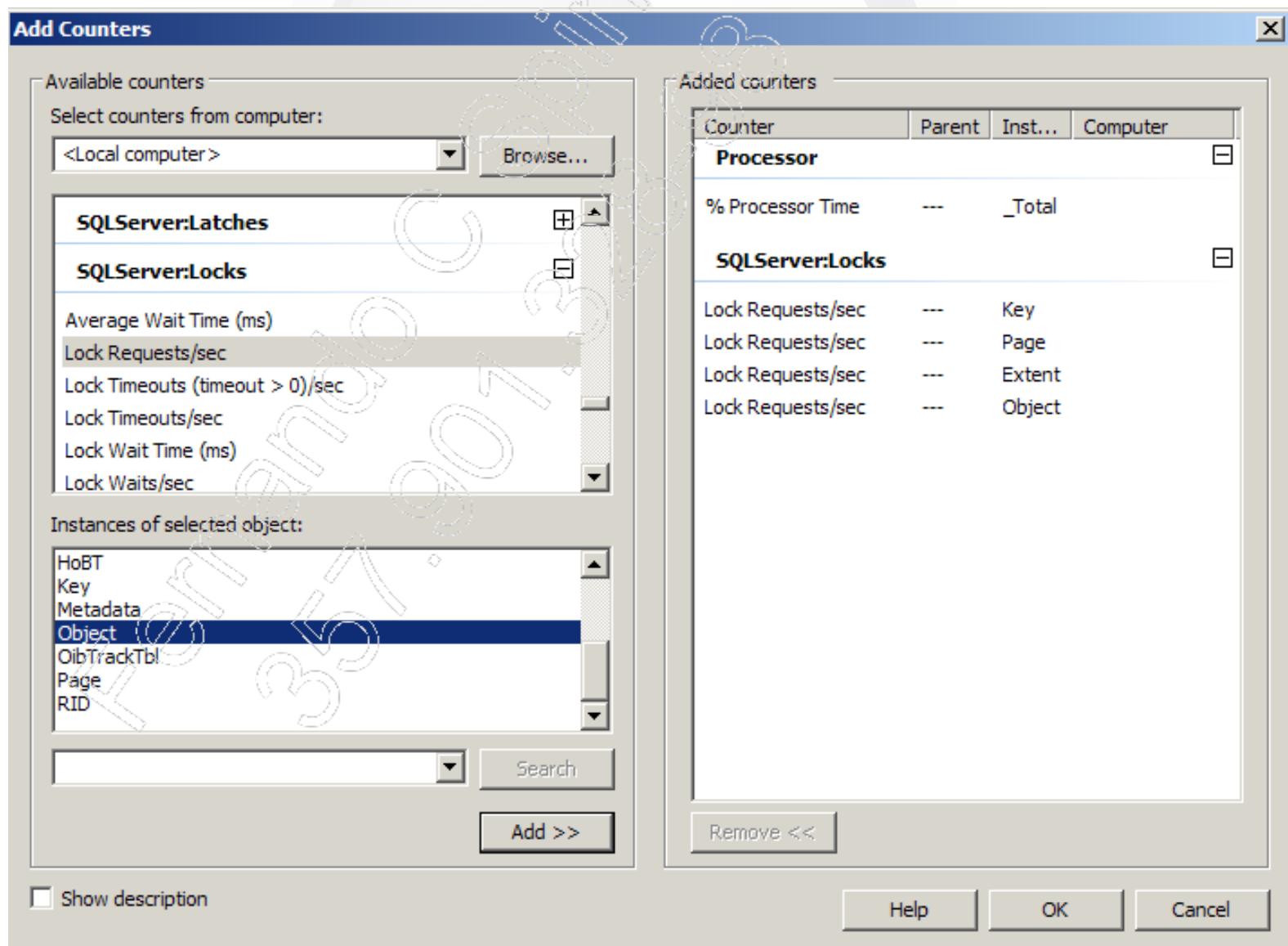


SQL 2014 - Módulo III

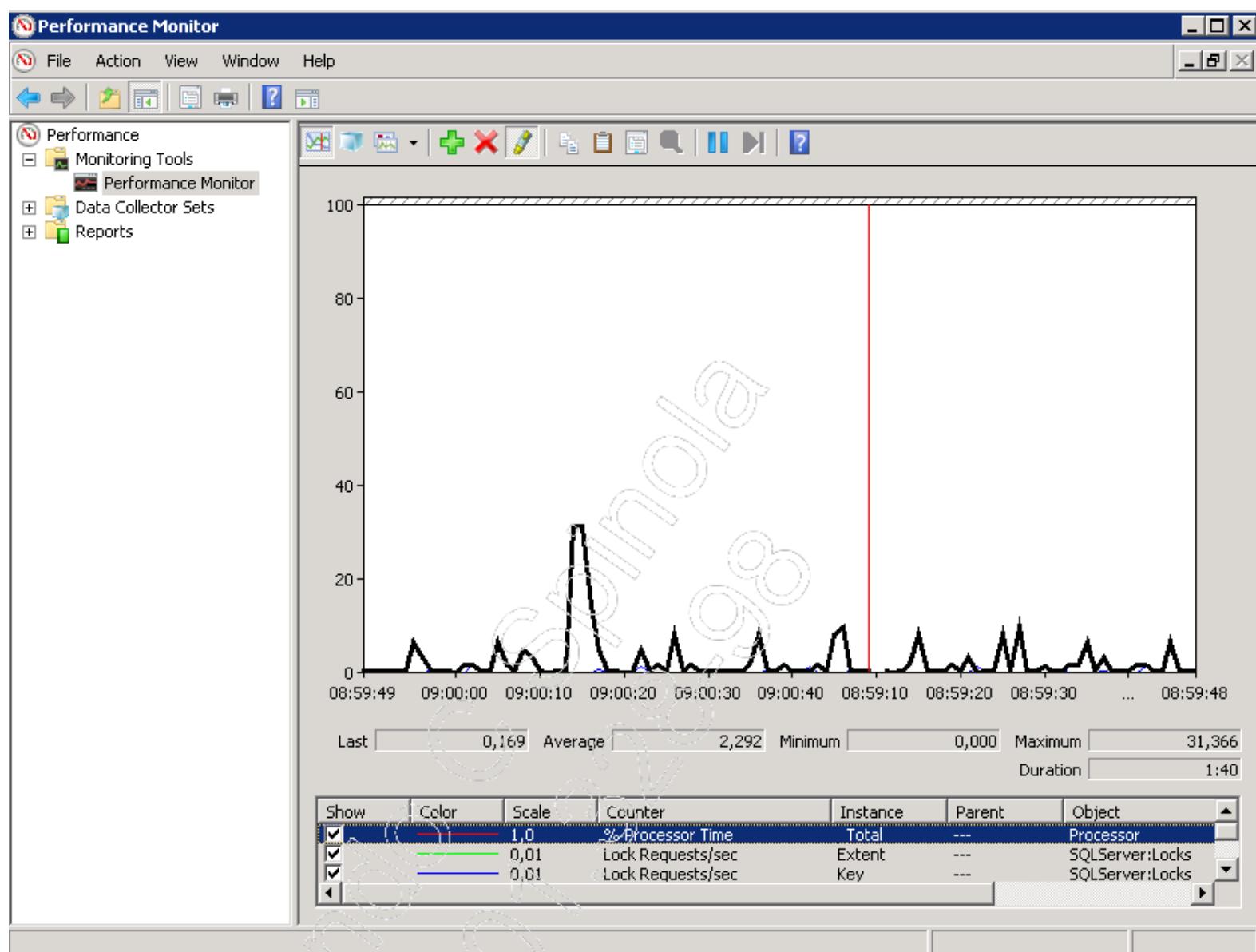
5. Para configurar os contadores, siga esta tabela:

Performance object	Counter	Instance	Clicar no botão
Processor	%Processor Time	_Total	Add
SQL:Locks	Lock Requests/sec	Extent	Add
		Key	Add
		Page	Add
		Object	Add

6. Após configurar os contadores como mostra a tabela anterior, clique no botão OK:



7. Na parte inferior da tela seguinte, observe os contadores. Clique sobre **%Processor Time** e, na barra de ferramentas, clique no ícone **Highlight**:



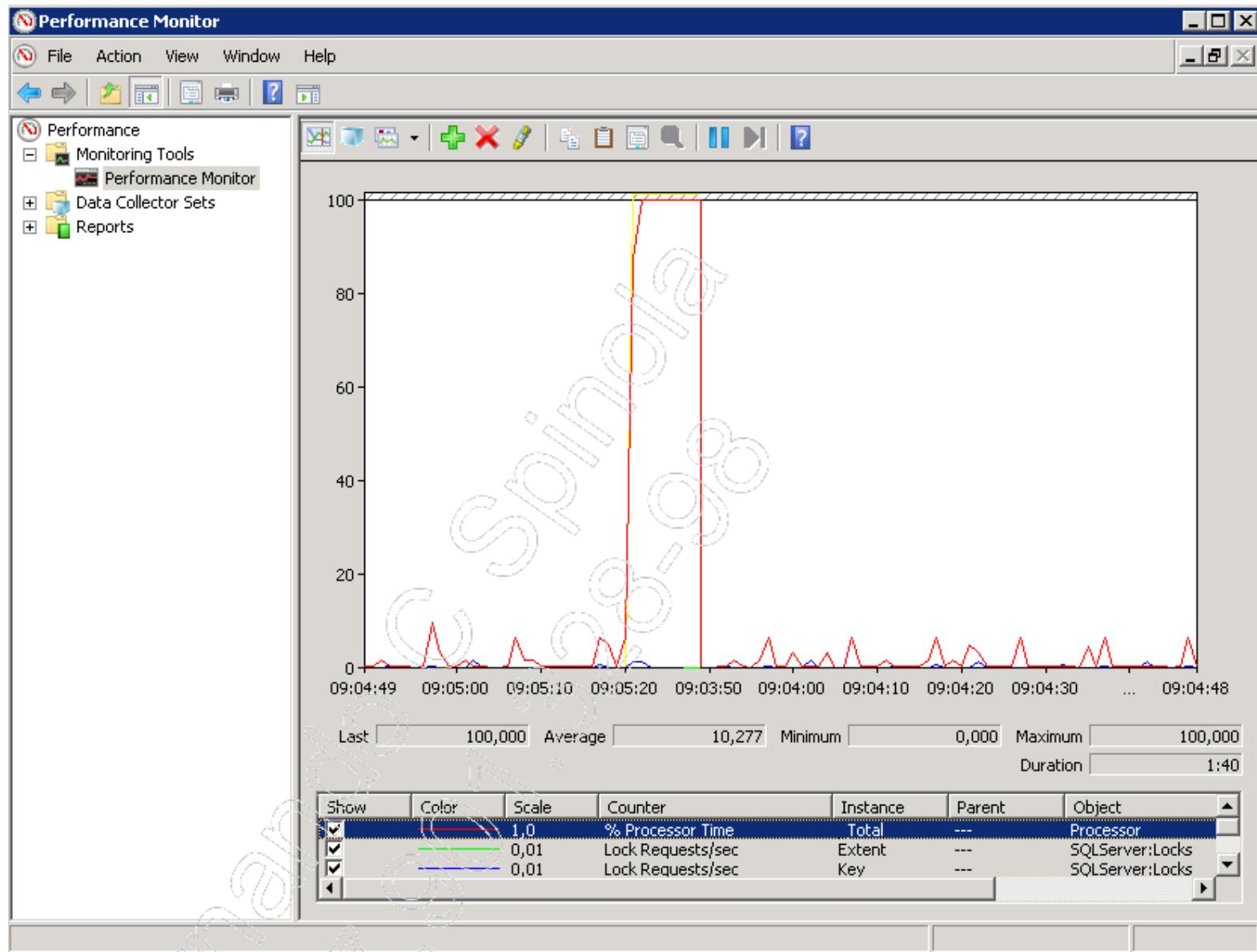
8. Observe que, ao clicar em **Highlight**, a linha do gráfico referente a este contador ficou preta para facilitar sua visualização;

9. Maximize o **SQL Server Management Studio** e, na barra de ferramentas, clique em **File**, **Open** e em **File** novamente. Conecte-se à primeira instância do SQL Server 2014 com a autenticação do Windows. Abra o **Script_02** da pasta **Capítulo_11** e pressione F5 para executá-lo;

SQL 2014 - Módulo III

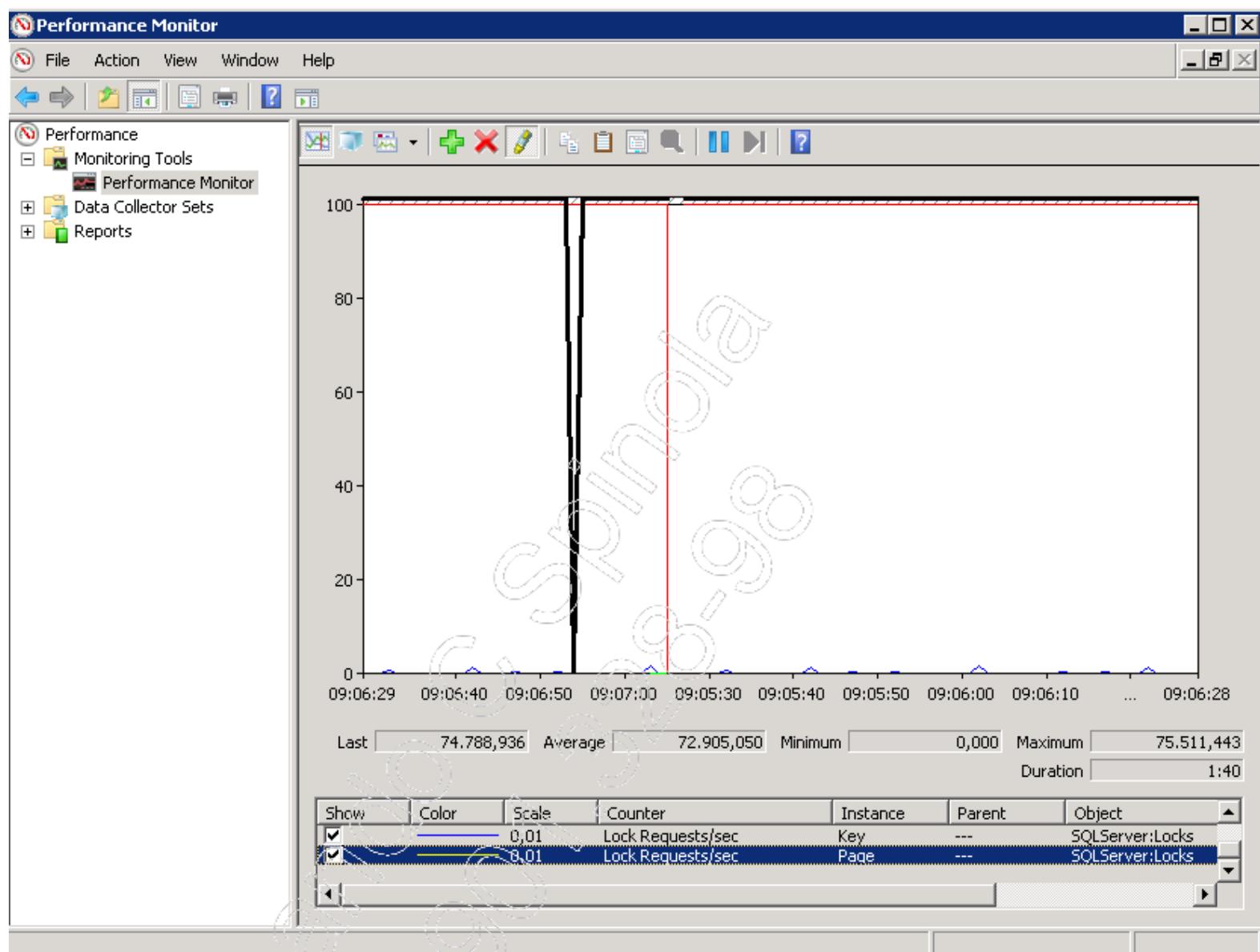
10. Deixe o script rodando e minimize o **SQL Server Management Studio**;

11. Observe, no **Performance Monitor**, o comportamento dos contadores:



12. Observe que o contador **%Processor Time** está marcando 100%, o que significa que ele está sendo muito exigido pela atividade do SQL Server (aquele a que o submetemos com o **Script_02**);

13. Clique no contador **Lock Request/sec - Page** e observe a linha referente a este contador no gráfico:

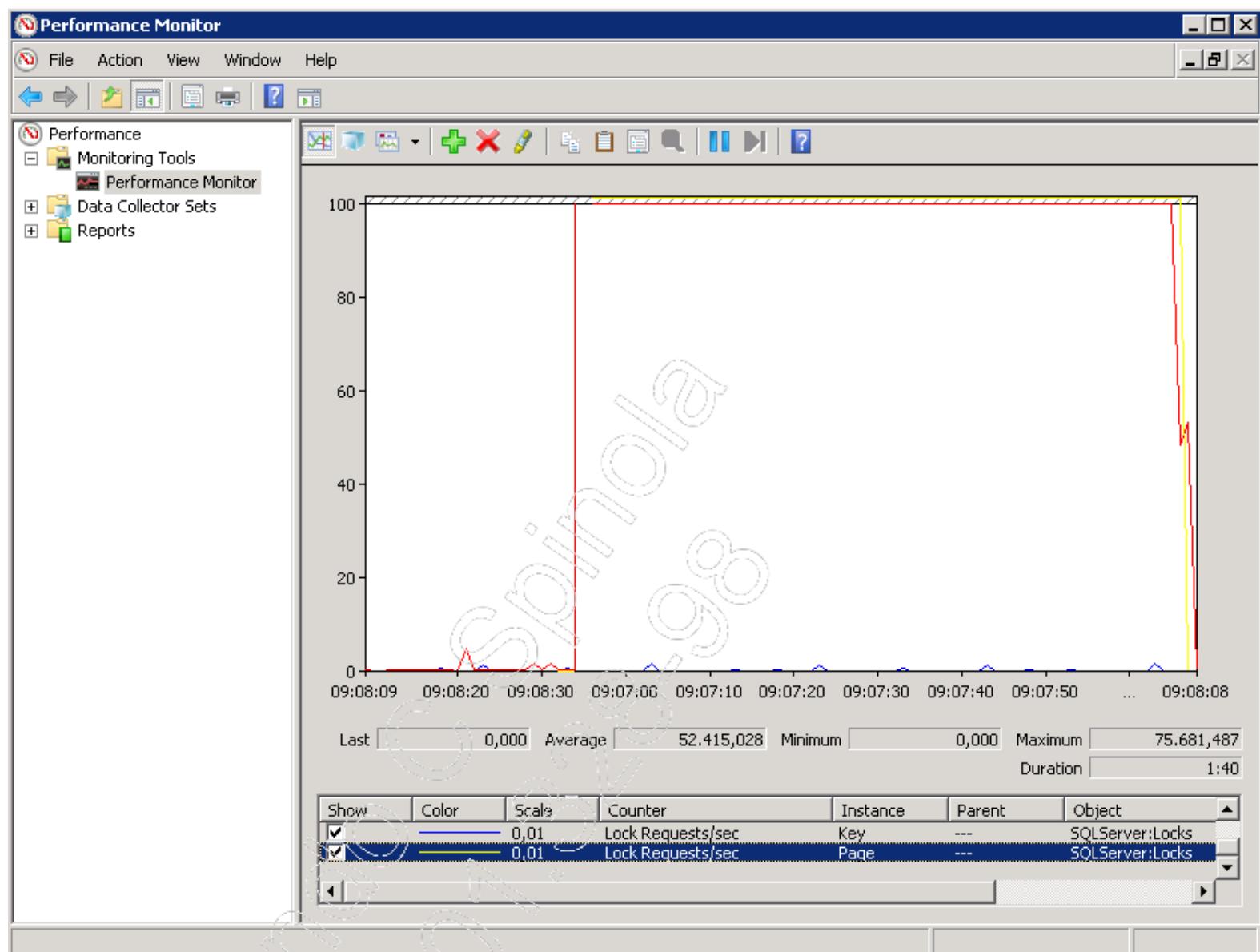


14. Repita o mesmo procedimento do passo anterior para os outros contadores do **Performance Monitor** desta configuração e observe que o SQL Server, para executar os comandos do **Script_02**, teve que colocar certos tipos de **Locks** nas respectivas tabelas;

15. Feche o **SQL Server Management Studio** para interromper o processamento;

SQL 2014 - Módulo III

16. Observe que, ao interromper o processamento do **Script_02**, os contadores que estavam sendo analisados no **Performance Monitor** caíram para 0%:



17. Feche o **Performance Monitor**.

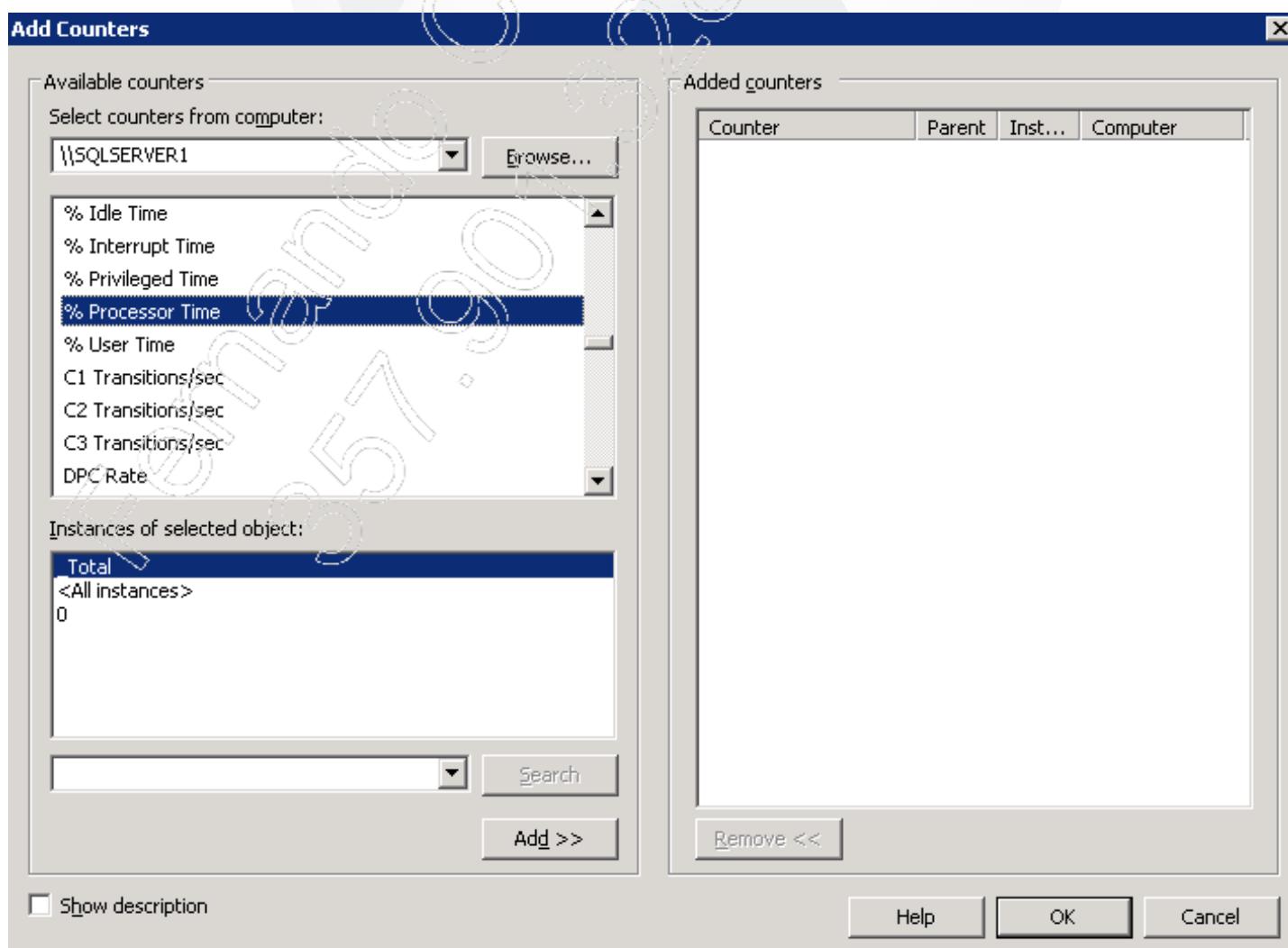
Laboratório 2



Neste exercício, vamos utilizar a preparação de ambiente do exercício anterior.

A – Monitorando com o Performance Monitor

1. Abra o **Performance Monitor**. Para tanto, clique no botão **Start**, escolha as opções **All Programs**, **Administrative Tools** e **Performance Monitor**;
2. Do lado direito da tela exibida, na barra de ferramentas, clique no sinal de adição (+) para adicionar contadores a serem monitorados;
3. Na tela seguinte, na opção **Select counters from computer**, deixe selecionado o nome do servidor atualmente usado. Em seguida, selecione alguns itens da listagem de objetos de performance, juntamente com alguns de seus respectivos contadores:



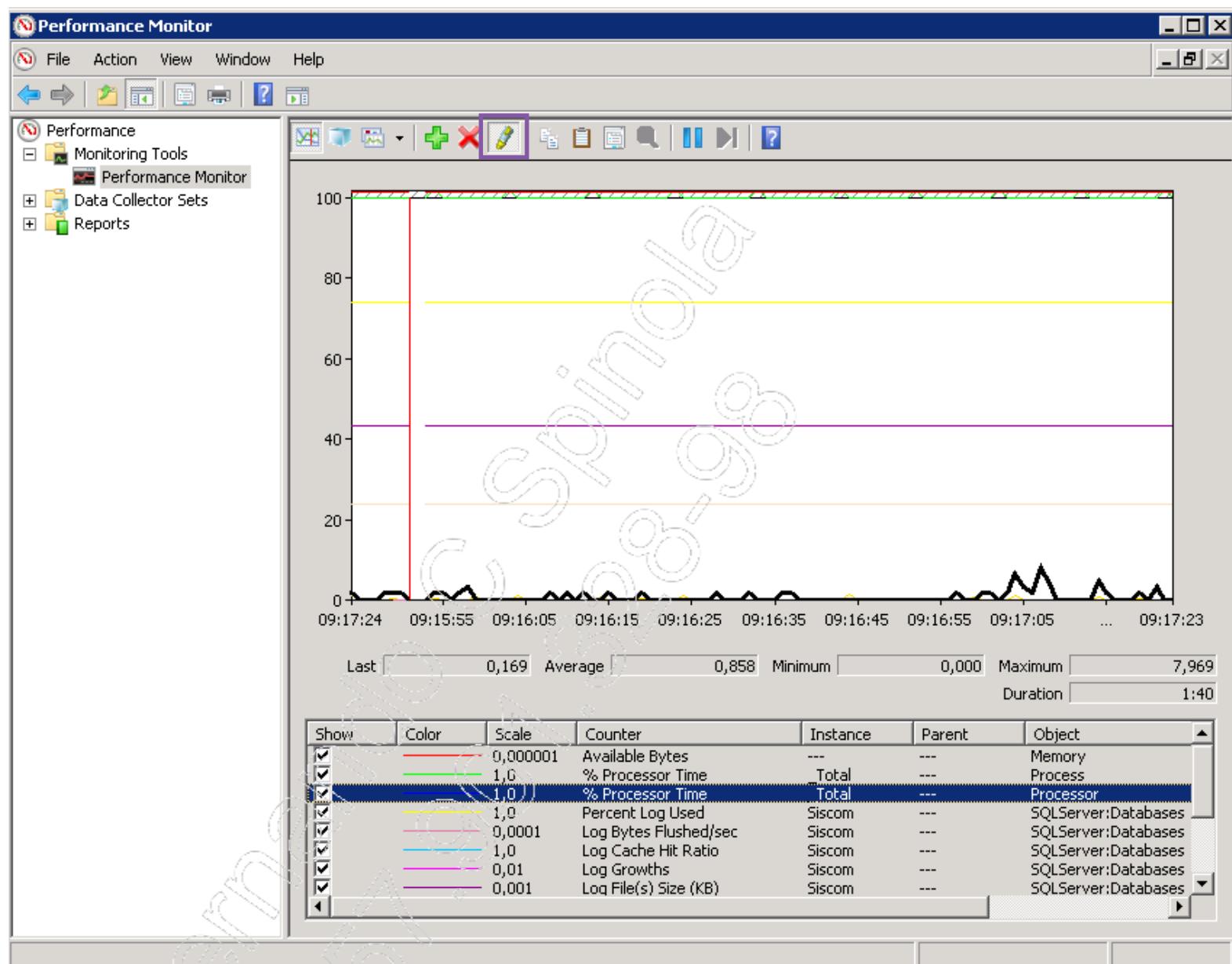
SQL 2014 - Módulo III

4. Para configurar o Performance Monitor, siga esta tabela:

Performance object	Counter	Instance	Clicar no botão
Memory	Available Bytes	--	Add
Process	%Processor Time	sqlservr	Add
Processor	%Processor Time	_Total	Add
SQL Server:Database	Percent Log Used	<all instances>	Add
	Log Bytes Flushed/sec	<all instances>	Add
	Log Cache Hit Ratio	<all instances>	Add
	Log Growths	<all instances>	Add
	Log File(s) Size (KB)	<all instances>	Add
SQL Server:Locks	Lock Requests/sec	Key	Add
		Page	Add
SQL Server:Memory Manager	Target Server Memory (KB)	--	Add
	Total Server Memory (KB)	--	Add

5. Após configurar os contadores conforme a tabela anterior, clique no botão OK;

6. Na parte inferior da seguinte tela, observe os contadores. Clique sobre **%Processor Time** do objeto **Processor** e, na barra de ferramentas, clique no ícone **Highlight**:



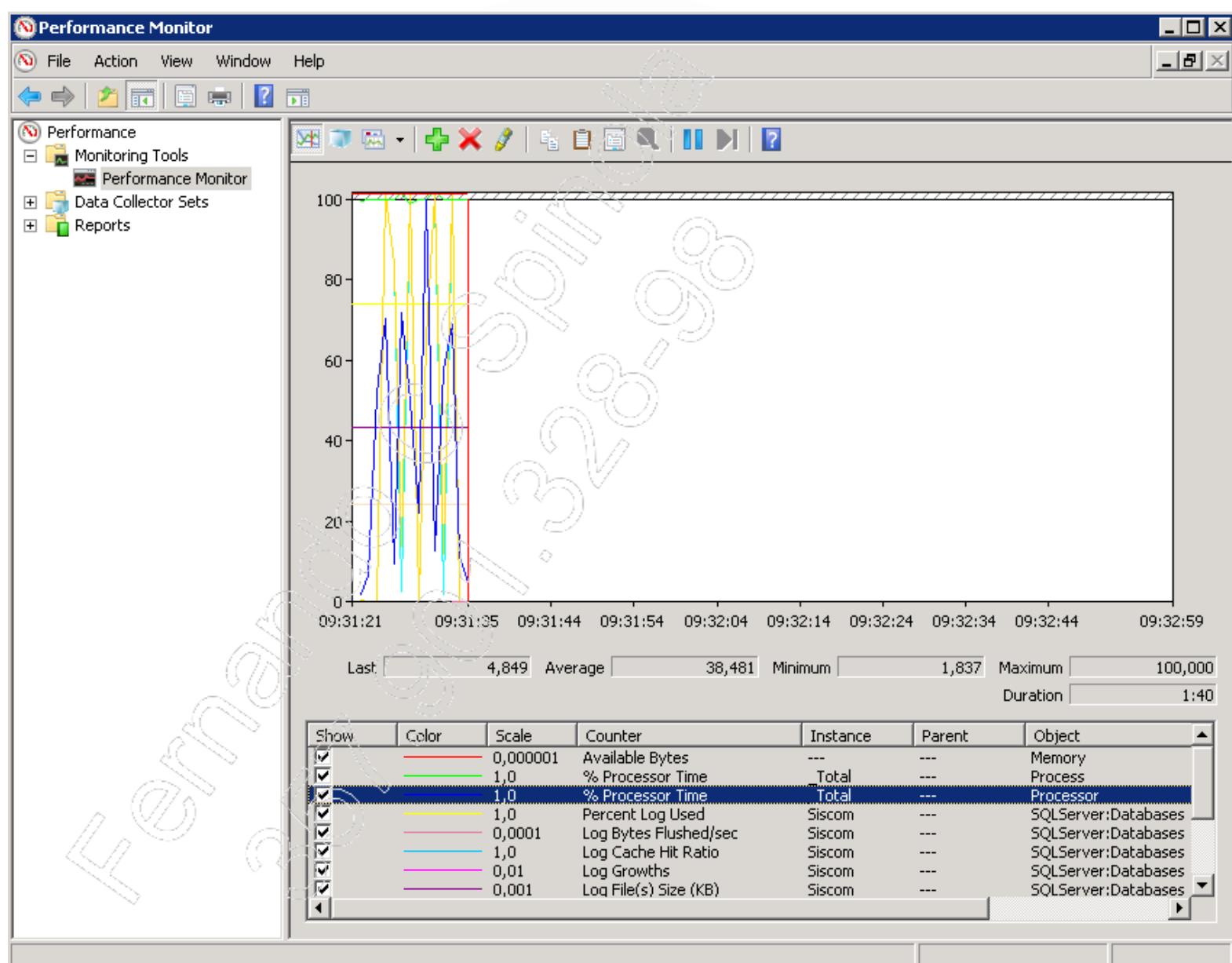
7. Observe que, ao clicar em **Highlight**, a linha do gráfico referente a este contador ficou preta para facilitar sua visualização;
8. Abra o **SQL Server Management Studio** conectando-se à primeira instância do **SQL Server 2014** com a autenticação do Windows;

SQL 2014 - Módulo III

9. Na barra de menus, clique sobre as opções **File** e **Open** e, em seguida, em **File** novamente. Conecte-se à primeira instância do SQL Server 2014 com a autenticação do Windows. Abra o **Script_03** do **Capítulo_11** e pressione F5 para executá-lo;

10. Deixe o script rodando e minimize o SQL Server Management Studio;

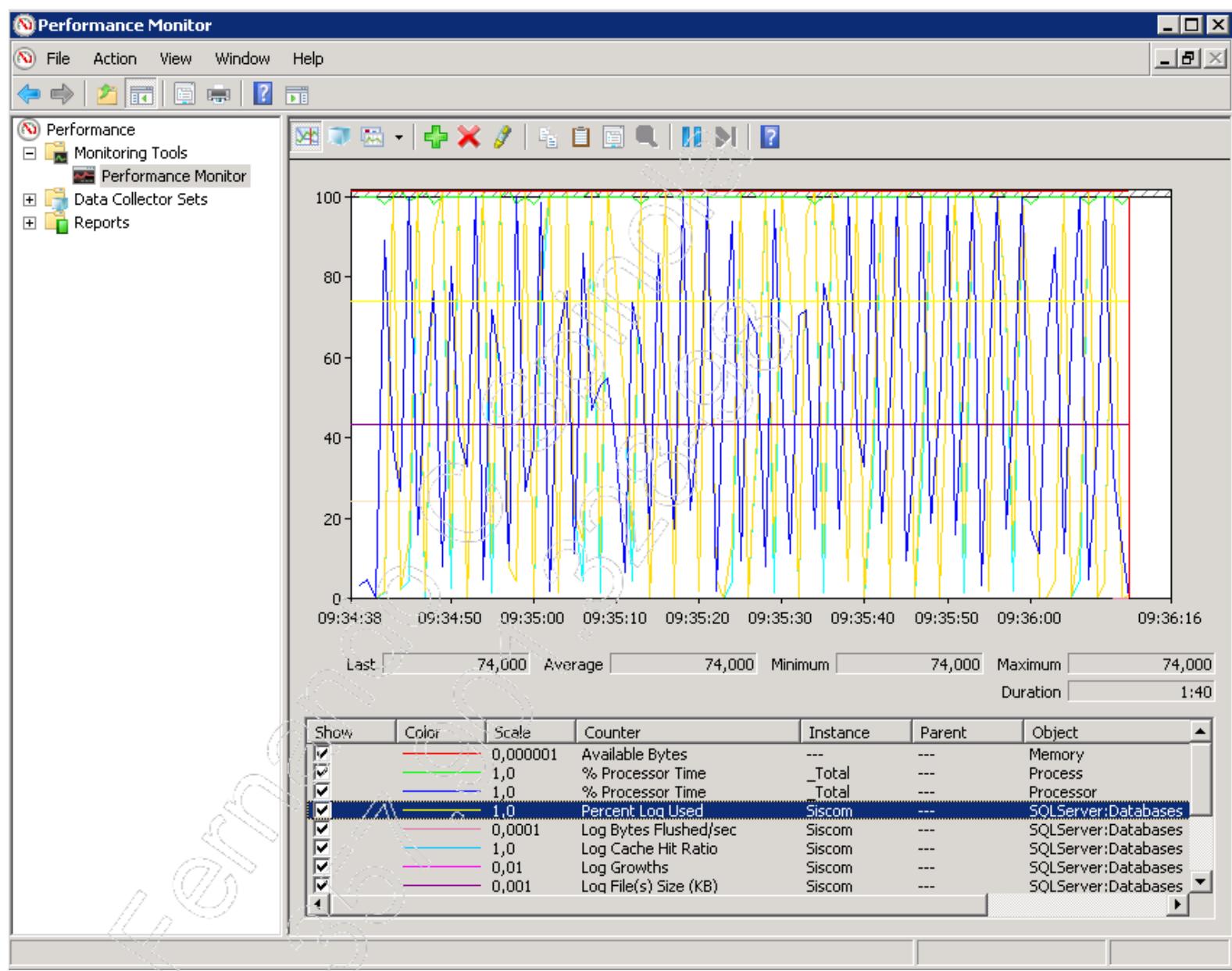
11. Observe, no **Performance Monitor**, o comportamento dos contadores:



12. Observe o comportamento deles na tela;

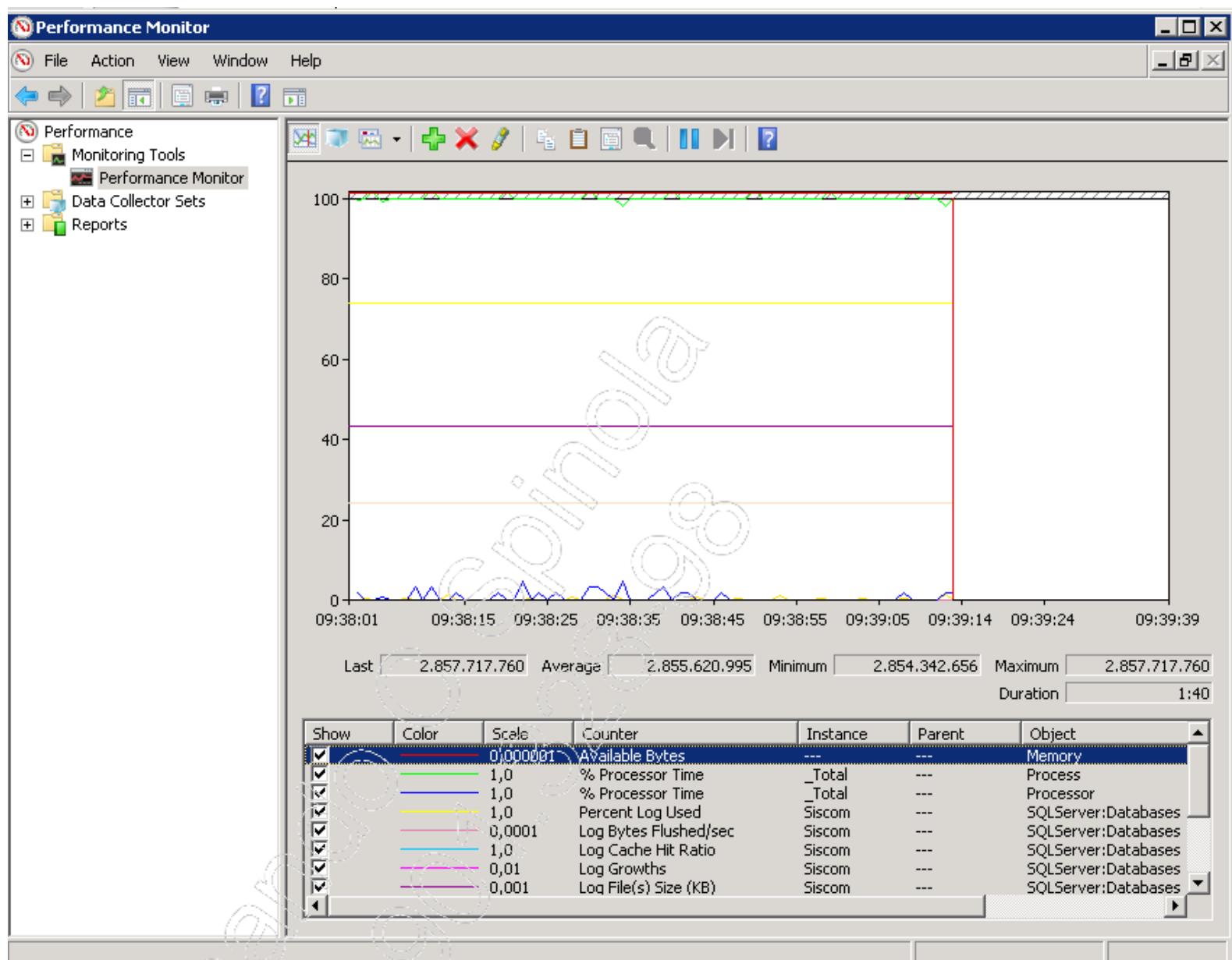
13. Feche o Script_03 para interromper o processamento;

14. Observe que, ao interromper o processamento da query do Script_03, alguns dos contadores do **Performance Monitor** caíram para 0% e os contadores **Log Percent Used** e **Log File(s) Used Size (KB)** se mantêm com uma porcentagem alta:



SQL 2014 - Módulo III

15. Aguarde alguns segundos sem realizar qualquer tipo de processamento até que o sistema se estabilize. Observe novamente o **Performance Monitor**:



16. Feche o **Performance Monitor**.

Laboratório 3

O objetivo deste exercício é monitorar uma atividade do SQL Server utilizando o **SQL Profiler**. Ele deverá ser feito em quatro partes:

- Criar um modelo (template) de um trace;
- Preparar a query a ser testada;
- Criar um trace com base no modelo criado anteriormente;
- Monitorar o banco de dados.



Neste laboratório, vamos utilizar a preparação de ambiente executada no **Laboratório 1**.

A – Criando um modelo (template) de um trace

1. Abra o **SQL Server Profiler**. Para tanto, clique no botão **Start** e escolha as opções **All Programs** e **Microsoft SQL Server 2014**. Em seguida, escolha **Performance Tools** e a opção **SQL Server Profiler**;
2. Na tela que será exibida, na barra de ferramentas, clique na opção **File** e, em seguida, escolha a opção **Templates** e depois a opção **New Template**;
3. Na tela que será exibida, no campo **New template name**, escreva **SQLProfiler_Plano_e_Duração** e clique na opção **Use as a default template for selected server type**;
4. Clique na guia **Events Selection**;

SQL 2014 - Módulo III

5. Na tela que será exibida, faça as seguintes configurações:

5.1. No campo **Events**, expanda a opção **Performance** e escolha as seguintes opções:

- **Auto stats;**
- **Showplan All;**
- **Showplan Text.**

5.2. Expanda a opção **TSQL** e escolha a opção **SQL:StmtCompleted**.

6. Clique no botão **Save**;

7. Minimize o **SQL Profiler**.

B – Preparando a query a ser testada

1. No **Service** do Windows, desligue o serviço **SQL Server Agent** para que o **SQL Profiler** possa monitorar apenas a query desejada;

2. Abra o **SQL Server Management Studio** e o **Script_04** da pasta **Capítulo_11**, conectando-se com a autenticação do Windows;

3. Deixe esse script aberto, mas não o execute ainda.

C – Criando um trace com base no modelo criado anteriormente

1. Maximize o **SQL Profiler**;

2. Na barra de menus, escolha a opção **File** e, em seguida, **New Trace**;

3. Conecte-se com a autenticação do Windows;

4. Na tela que será exibida, no campo **Trace name**, escreva **Filtro_Comercio**;
5. Observe que, no campo **Use the template**, está escrito **SQLProfiler_Plano_e_Duração**;
6. Clique no botão **Run**.

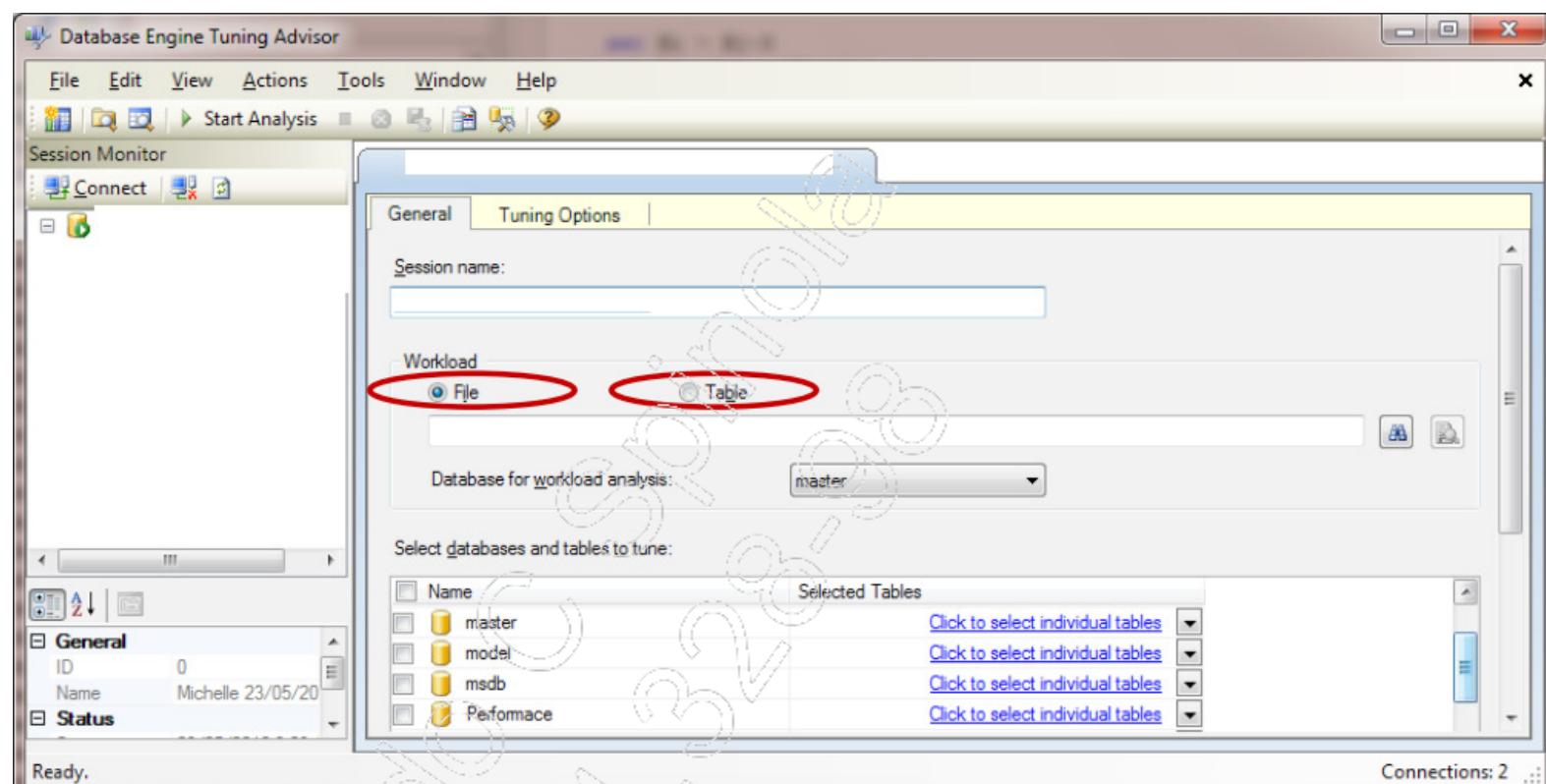
D – Monitorando o banco de dados

1. Expanda o **SQL Server Management Studio** e pressione F5 para executar o **Script_04**;
2. Minimize o **SQL Server Management Studio** novamente e maximize o **SQL Profiler**;
3. No **SQL Profiler**, na coluna **EventClass**, mova a barra de rolagem até encontrar a última linha, na qual está escrito **SQL:StartCompleted**, e observe, na parte inferior da tela, a query que foi executada;
4. Selecione em uma linha acima a opção **Showplan Text** e observe, na parte inferior da tela, o plano de execução da query executada no **Query Analyzer**;
5. No meio desta tela, movimente a barra de rolagem para a direita até encontrar a coluna **Reads** e observe a quantidade de leitura feita pelo **SQL Server** para executar esta query. Observe a quantidade de linhas lidas ao executar esta query;
6. Feche o **SQL Profiler**;
7. Feche o **SQL Server Management Studio**.

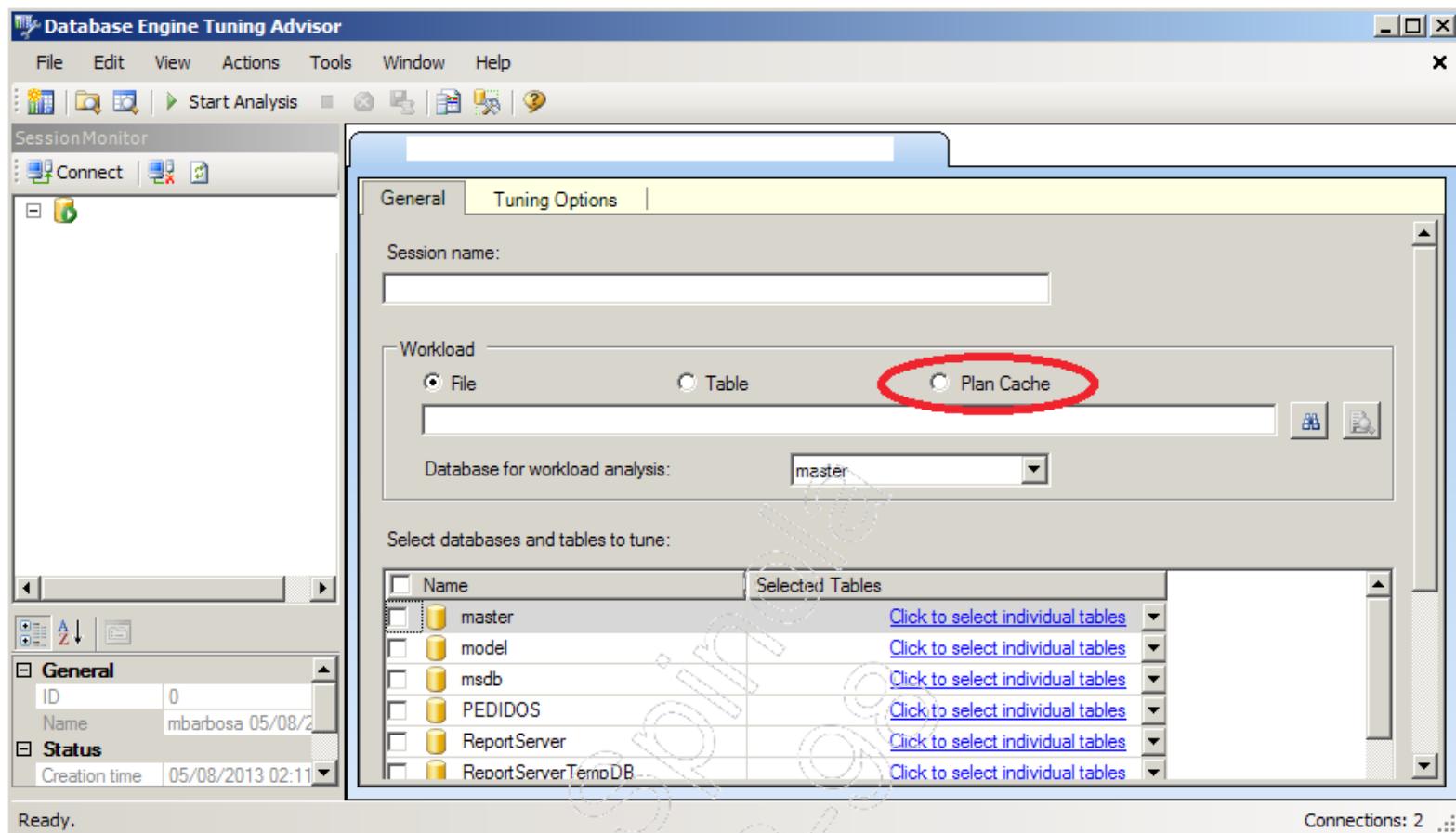
Laboratório 4

A - Monitorando a performance com o Database Profiler e o Database Engine Tuning Advisor

- Database Engine Tuning Advisor Versão 2014



- Database Engine Tuning Advisor Versão 2014



1. No SQL Server Management Studio, crie um novo banco de dados chamado BancoPerformance (Script_04);

2. Crie a tabela a seguir:

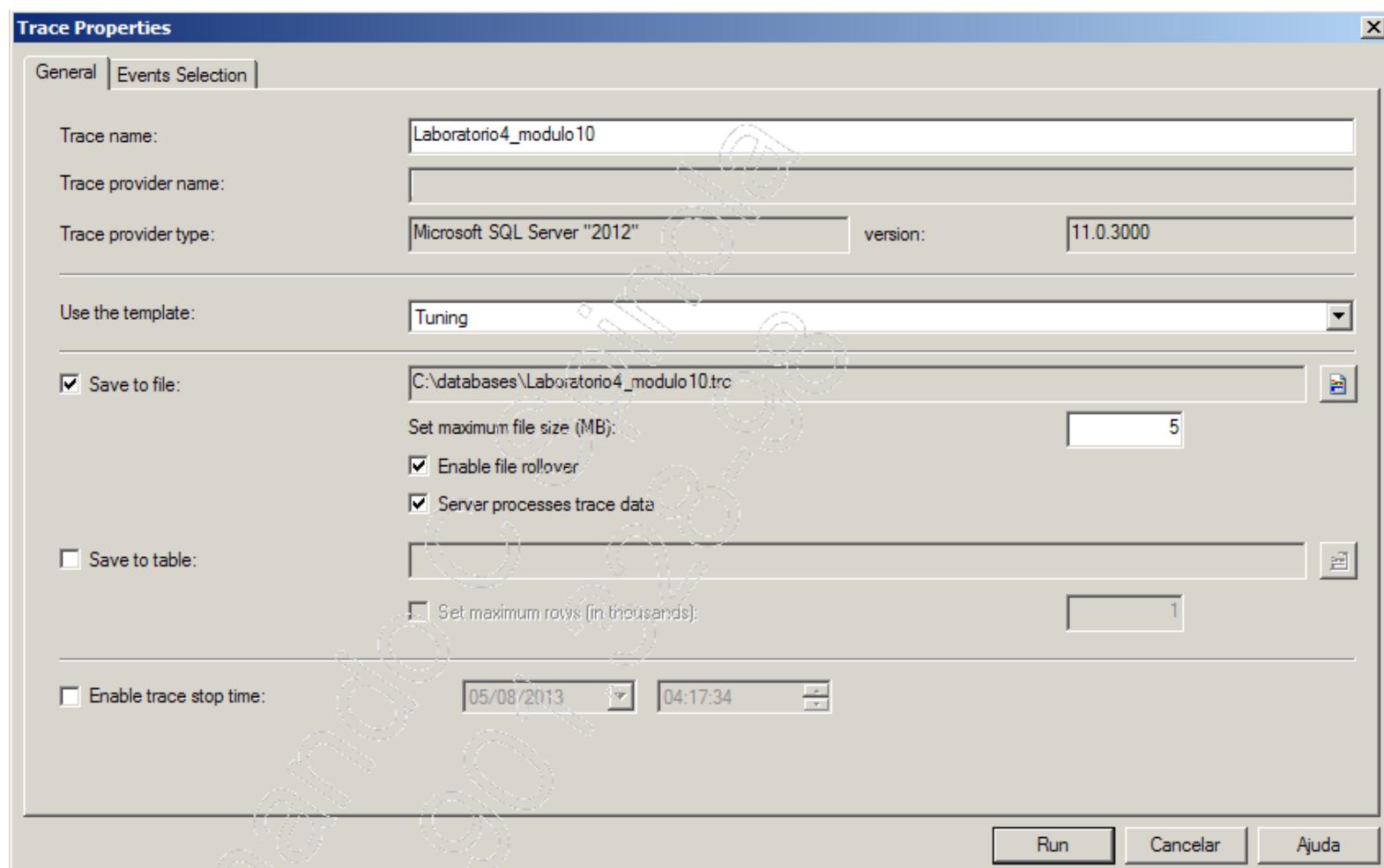
```
CREATE TABLE tblErro (
    NumErroID BIGINT IDENTITY PRIMARY KEY,
    MensagemErro VARCHAR(100),
    CodErro INT,
    Linguagem INT
)
GO
```

```
INSERT tblErro
select
    LEFT([Text],20),
    severity,
    language_id
FROM Sys.messages
```

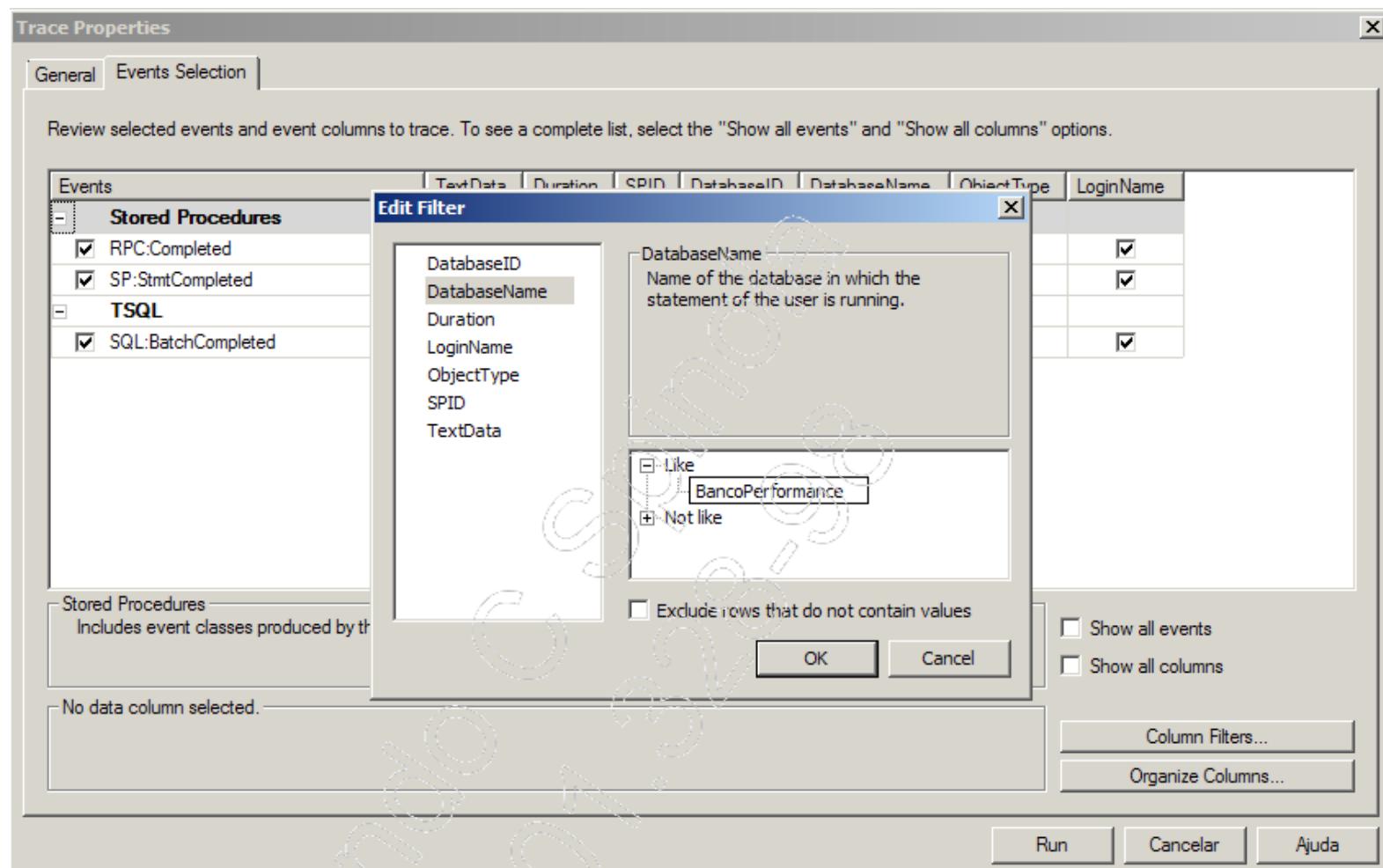
SQL 2014 - Módulo III

3. Abra o **SQL Profiler**. Para tanto, clique no botão **Start**, escolha as opções **All Programs, SQL Server 2014, Performance Tools e SQL Server Profiler**. Outra alternativa é utilizar o menu **tools** do **SQL Server Profiler**;

4. Informe as opções a seguir:



5. Selecione a pasta **Events Selection**, clique no botão **Column Filters**, selecione a opção **Database Name** e selecione a opção **Like**. Em seguida, digite **Performance**:



6. Clique em **OK**;
7. Selecione a opção **Run**;

SQL 2014 - Módulo III

8. Retorne para o SSMS e realize as seguintes consultas três vezes:

```
SELECT MensagemErro, CodErro
FROM tblErro
WHERE codErro = 16;
SELECT MensagemErro, CodErro
FROM tblErro
WHERE MensagemErro LIKE 'a%';
```

```
SELECT *
FROM tblErro
WHERE Linguagem > 1000;
```

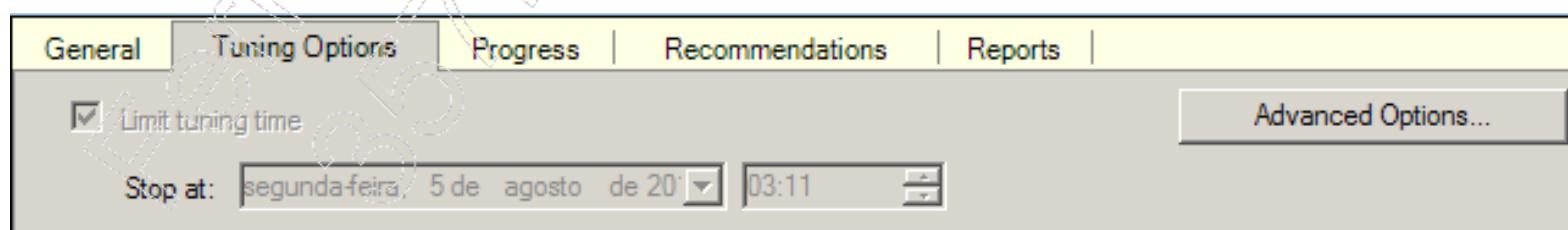
9. Retorne ao **SQL Profiler** e confira o resultado obtido:

EventClass	TextData	Duration	SPID	DatabaseID	DatabaseName	ObjectType	LoginName
SQL:BatchCompleted	SELECT MensagemErro, CodErro FROM ...	6211	52	9	BancoPerf...		WIN-OM...
SP:StmtCompleted	select table_id, item_guid, opIsn_f...	0	14	9	BancoPerf...	20816 - PQ	sa
SQL:BatchCompleted	SELECT MensagemErro, CodErro FROM ...	6469	52	9	BancoPerf...		WIN-OM...
SP:StmtCompleted	select table_id, item_guid, opIsn_f...	0	19	9	BancoPerf...	20816 - PQ	sa
SQL:BatchCompleted	SELECT MensagemErro, CodErro FROM ...	6373	52	9	BancoPerf...		WIN-OM...

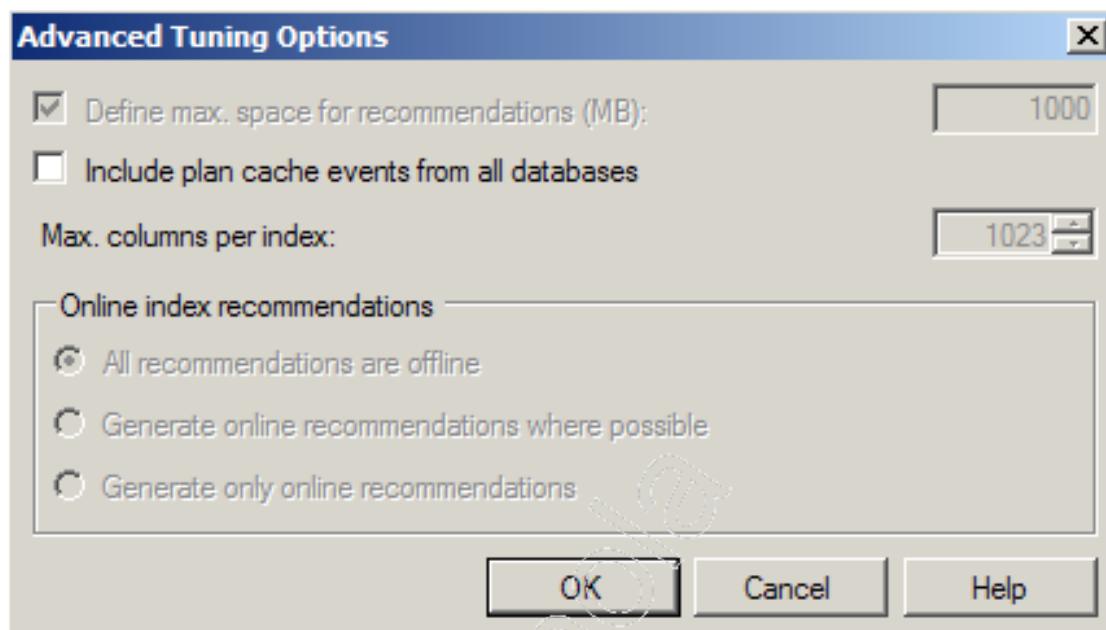
10. Feche a janela e confirme a gravação do arquivo de trace;

11. Abra o **Database Engine Tuning Advisor**. Para tanto, clique no botão **Start**, escolha as opções **All Programs, SQL Server 2014, Performance Tools e Database Engine Tuning Advisor**;

12. Selecione a pasta **Tuning Options** e clique no botão **Advanced Options...**:



13. Defina o valor 1000 na opção **Define max. space for recommendations(MB)**:



14. Selecione o botão **OK** e retorne para a pasta **General**;

15. Selecione o database **BancoPerformance**;

16. Abra a opção **File** e selecione o arquivo de trace gerado no SQL Profiler;

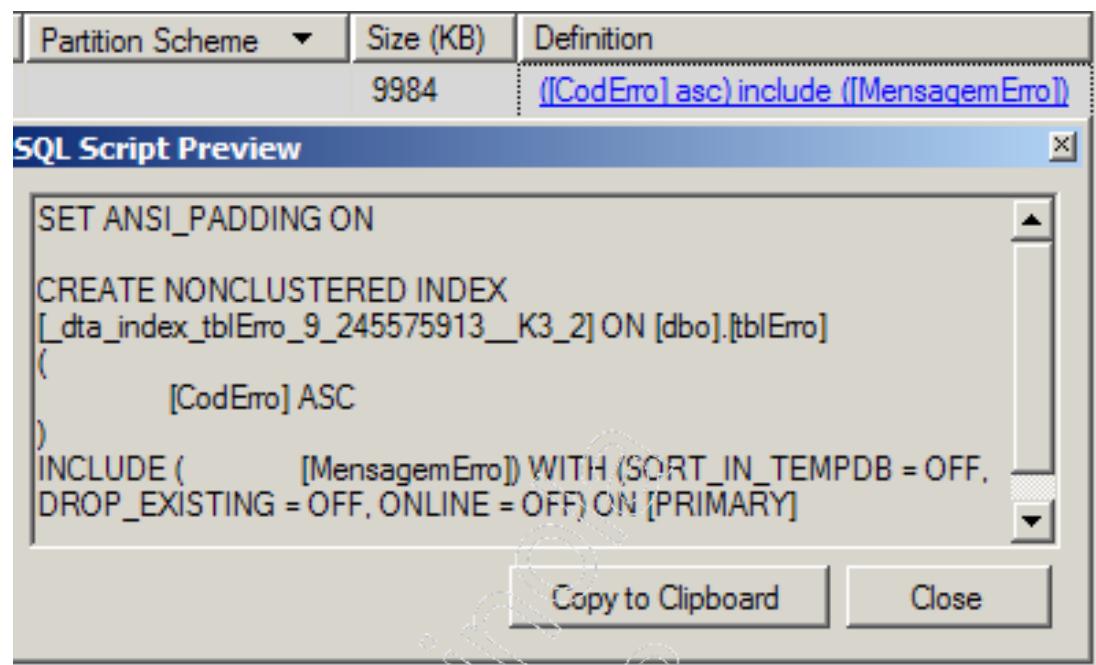
17. Clique no botão **Start analysis**;

18. Ao final, selecione a pasta **Recommendations** e verifique as recomendações obtidas:

Database Name	Object Name	Recommendation	Target of Recommendation	Details	Partition Scheme
BancoPerformance	[dbo].[tblEro]	create	_dta_index_tblEro_9_245575913_K3_2		
BancoPerformance	[dbo].[tblEro]	create	_dta_index_tblEro_9_245575913_K2_3		

SQL 2014 - Módulo III

19. Clique na coluna **Definition** e verifique a recomendação indicada:



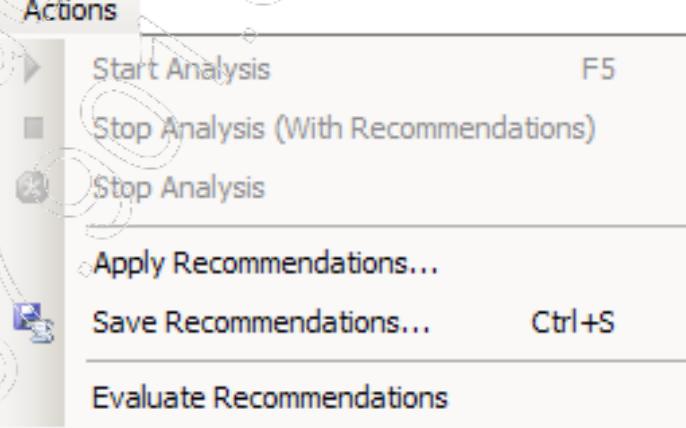
The screenshot shows a 'SQL Script Preview' window. At the top, there's a table with columns: Partition Scheme, Size (KB), and Definition. The 'Definition' column contains the recommended T-SQL script:

```
SET ANSI_PADDING ON  
  
CREATE NONCLUSTERED INDEX [_dta_index_tblEro_9_245575913_K3_2] ON [dbo].[tblEro]  
(  
    [CodEro] ASC  
)  
INCLUDE ([MensagemEro]) WITH (SORT_IN_TEMPDB = OFF,  
DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
```

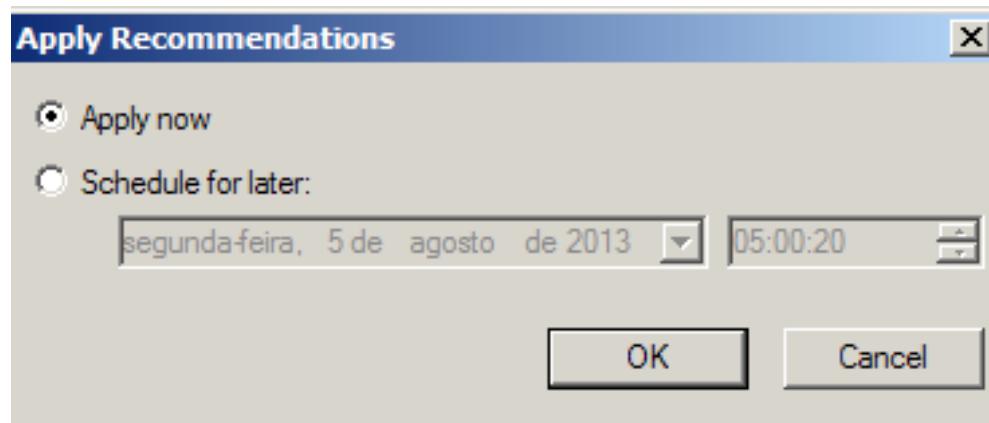
At the bottom of the window are two buttons: 'Copy to Clipboard' and 'Close'.

20. Selecione o menu **Actions**:

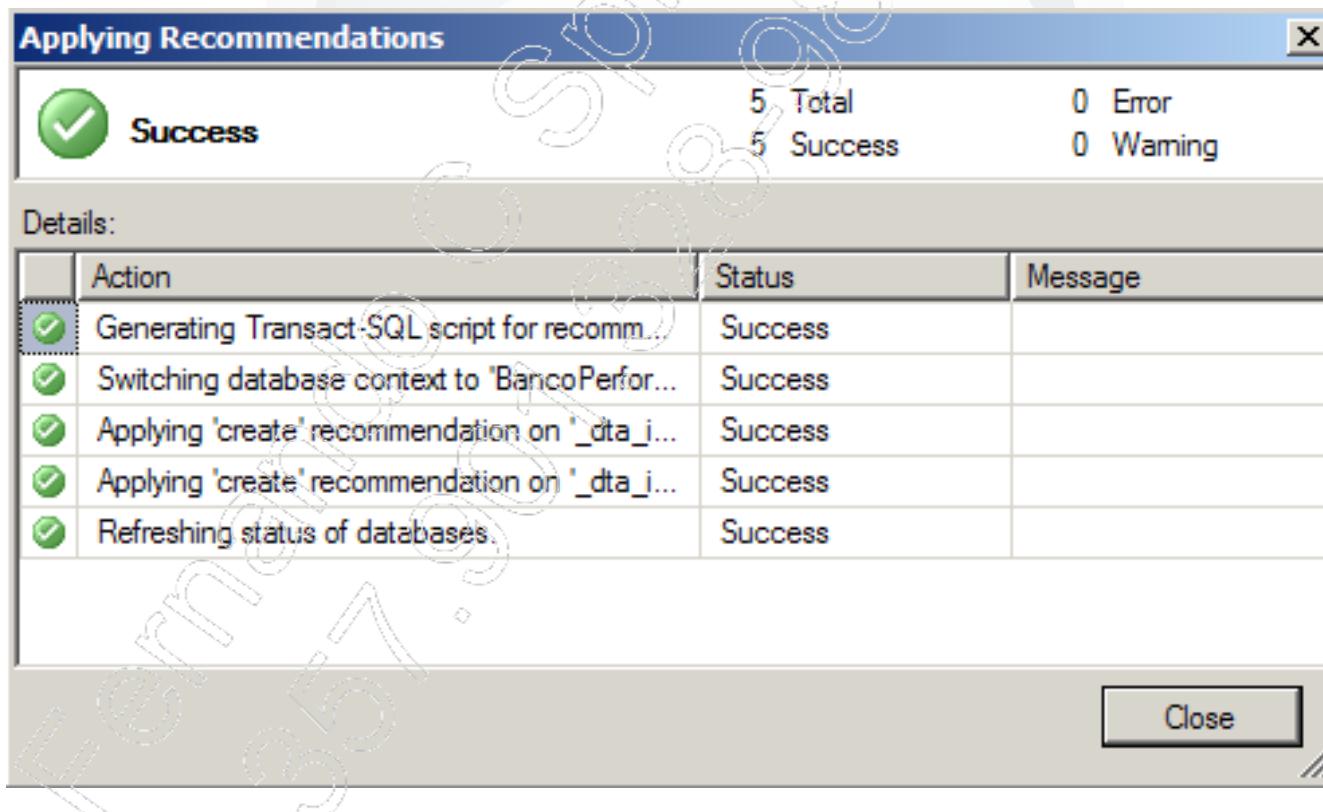
21. Selecione a opção **Apply Recommendations...**:



22. Selecione a opção **Apply now**:



23. Verifique se as recomendações indicadas foram realizadas e clique no botão **Close**:



12

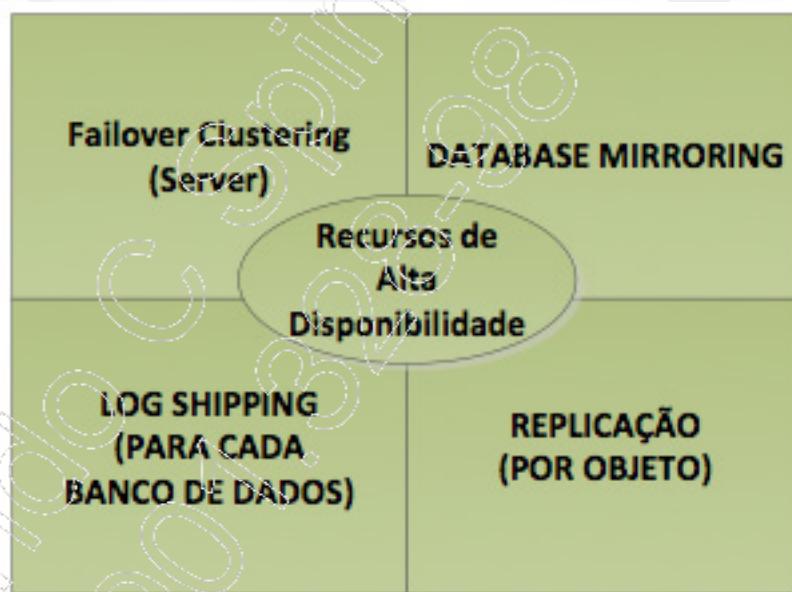
Alta disponibilidade

- ✓ Log Shipping;
- ✓ Database Mirroring;
- ✓ Always ON - SQL 2014.

12.1. Introdução

A alta disponibilidade (High Availability - HA) pode ser implementada através de vários recursos disponíveis no SQL Server, por exemplo, o **Log Shipping** para replicação de banco de dados (existente desde a versão 2000). Outro exemplo de recurso de alta disponibilidade é o **Database Mirroring**, no qual o banco de dados sofre atualização contínua e, em caso de falha do nó principal, o nó de standby pode assumir o papel do servidor primário e continuar o funcionamento do banco de dados. Esses dois recursos serão tratados com mais detalhes ao longo deste capítulo.

A seguir, veja os recursos de alta disponibilidade disponíveis no SQL Server até a versão 2008:



O SQL Server 2014 introduziu várias mudanças referentes à alta disponibilidade. A partir da versão 2014, os recursos de alta disponibilidade foram integrados aos de recuperação de desastres (Disaster Recover - DR) e essa integração recebeu o nome de **Always ON**.

O **Always ON** surgiu como uma alternativa ao uso do **Database Mirroring** e tornou-se o principal caminho adotado para a alta disponibilidade. Neste capítulo, conheceremos o funcionamento da alta disponibilidade em SQL Server utilizando o recurso **Always ON**.

12.2. Log Shipping

Log Shipping é um recurso disponível no SQL Server que permite que um banco de dados possa ser copiado para outro de forma a produzir uma cópia atualizada conforme o tempo configurado para essa operação (por exemplo, cinco minutos). Com isso, em caso de necessidade, podemos utilizar o servidor de cópia para realizar consultas e, em caso de falha da instância principal, a instância secundária ou de **Log Shipping** poderá ser ativada para uso imediato em caso de desastre.

Este recurso possui algumas características semelhantes à replicação de dados, porém, as diferenças são bastante claras e estão listadas a seguir:

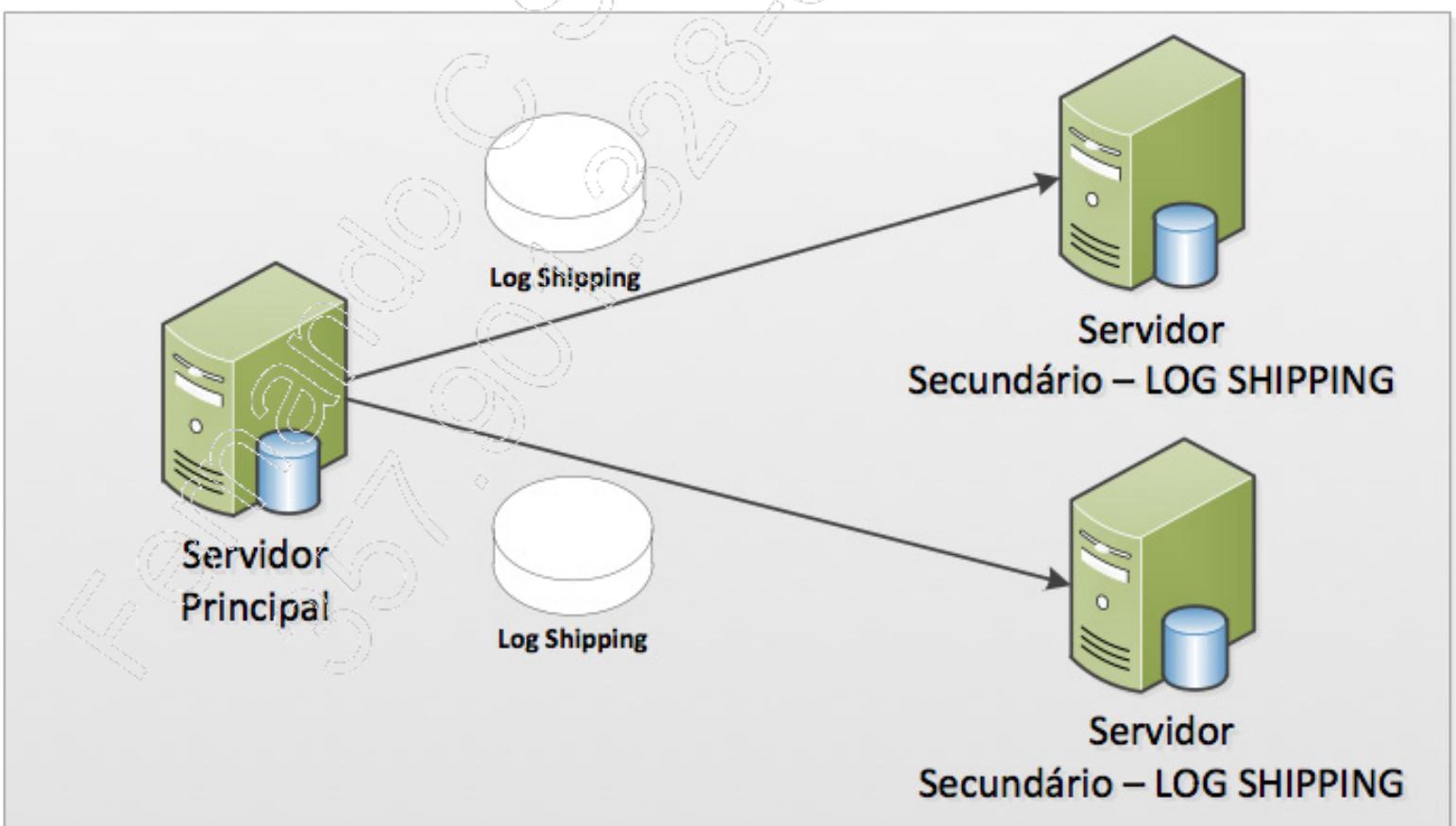
Característica	Log Shipping	Replicação
Latência	≥ 1 min	Segundos
Necessidade de chaves primárias nas tabelas envolvidas	Não	Sim
Possibilidade de envio individual de tabelas ou partes de tabelas	Não	Sim
Possibilidade de atuar como servidor principal	Sim	Não
Dados dos bancos de sistema são atualizados	Sim, a maior parte dos dados	Não
Foco de utilização	Recuperação de desastres (Disaster Recover -DR)	Alta disponibilidade (High Availability - HA)

SQL 2014 - Módulo III

A seguir, temos o diagrama representando um modelo de **Log Shipping** básico:



A seguir, vemos o diagrama representando um modelo de **Log Shipping** utilizando redundância (dois servidores secundários):



Para configuração do **Log Shipping**, são necessários os seguintes recursos:

- Pelo menos dois servidores com SQL Server Enterprise instalado, cada um com uma instância SQL Server;
- Conectividade de rede entre as instâncias, preferencialmente redes de alta velocidade;
- Compartilhamentos de diretórios entre os servidores;
- Banco de dados ativado como modo de recuperação **Full** (desejado) ou **Bulked Logged**;
- SQL Server Agent configurado e ativo nos dois servidores.

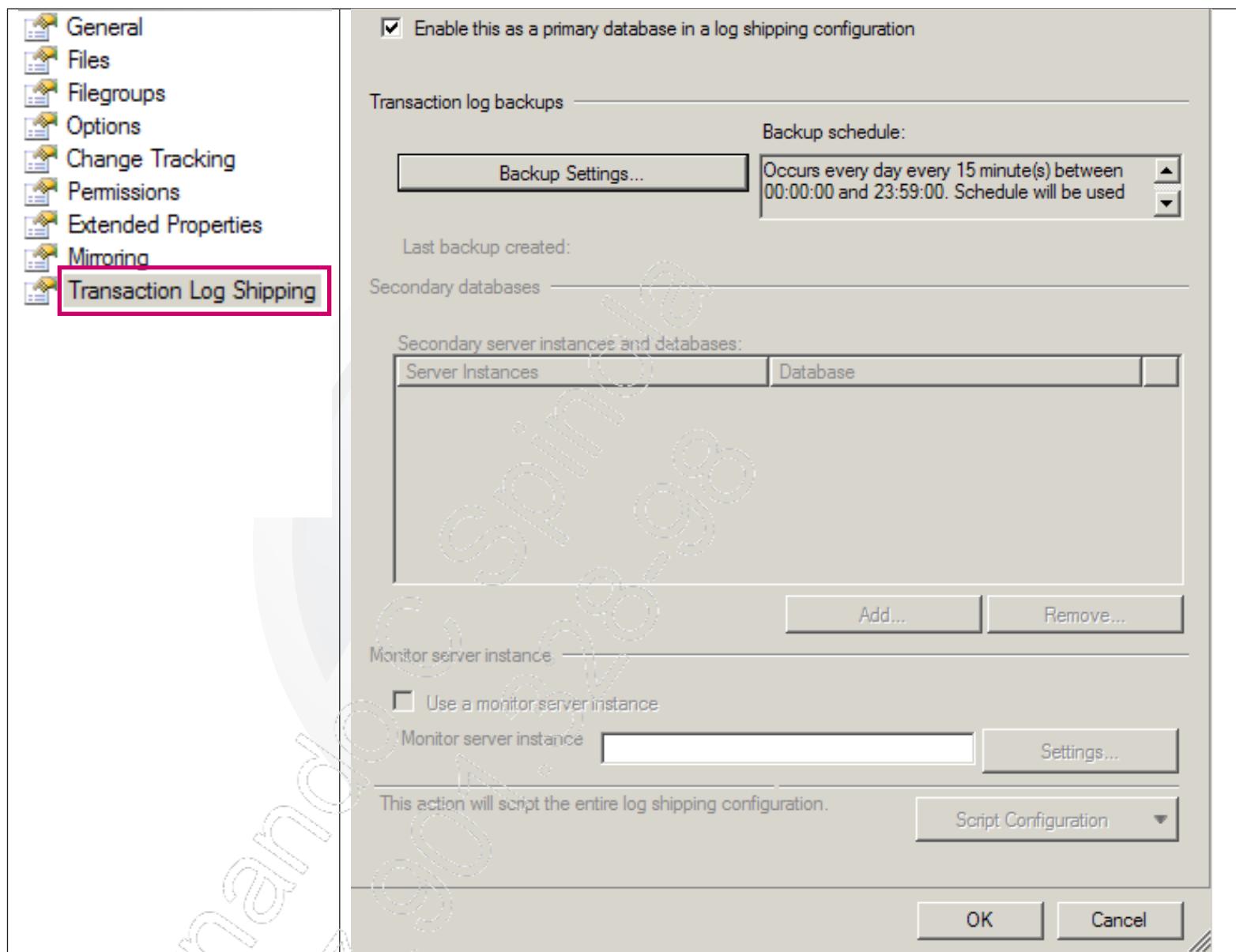
12.2.1. Configurando o Log Shipping

A seguir, vamos realizar a configuração do **Log Shipping**, utilizando seu assistente de configuração.

1. Defina o modo de recuperação do banco de dados do servidor primário como modo de recuperação completo (**Full**):

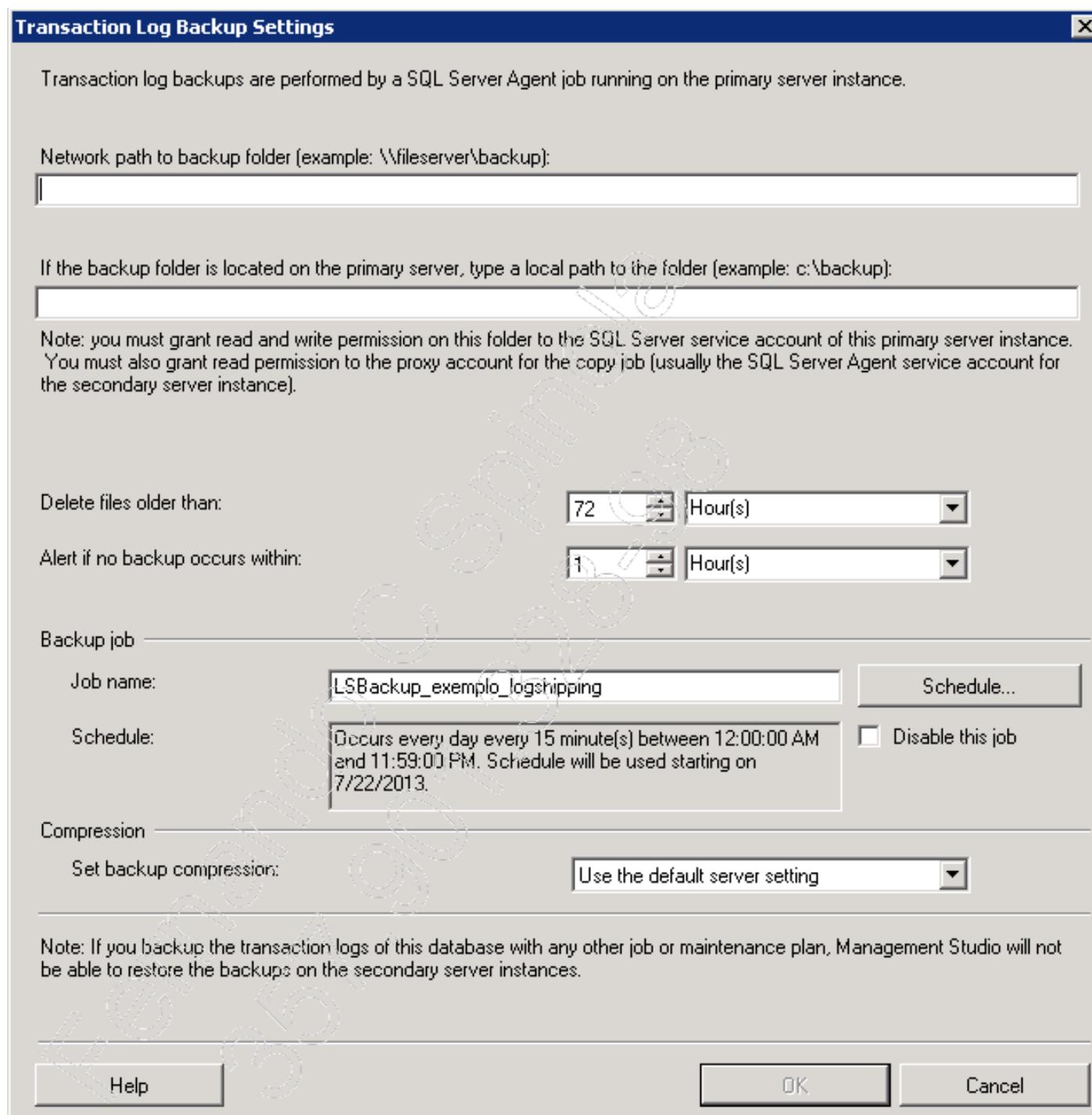
```
USE [master]
GO
ALTER DATABASE nome_db SET RECOVERY FULL WITH NO_WAIT
GO
```

2. Selecione o banco de dados desejado, pressione o botão direito do mouse e selecione a opção **Properties**. Será exibida a tela a seguir. Então, selecione a opção **Transaction Log Shipping** na lateral esquerda:



3. Selecione a opção **Enable this as a primary database in a log shipping configuration**. Ela indica que esse banco de dados será a instância do servidor primário para configuração do Log Shipping;

4. Pressione o botão **Backup Settings** para configuração de como será feito o backup dos logs para envio ao servidor secundário. A tela a seguir será exibida para configuração de backup:

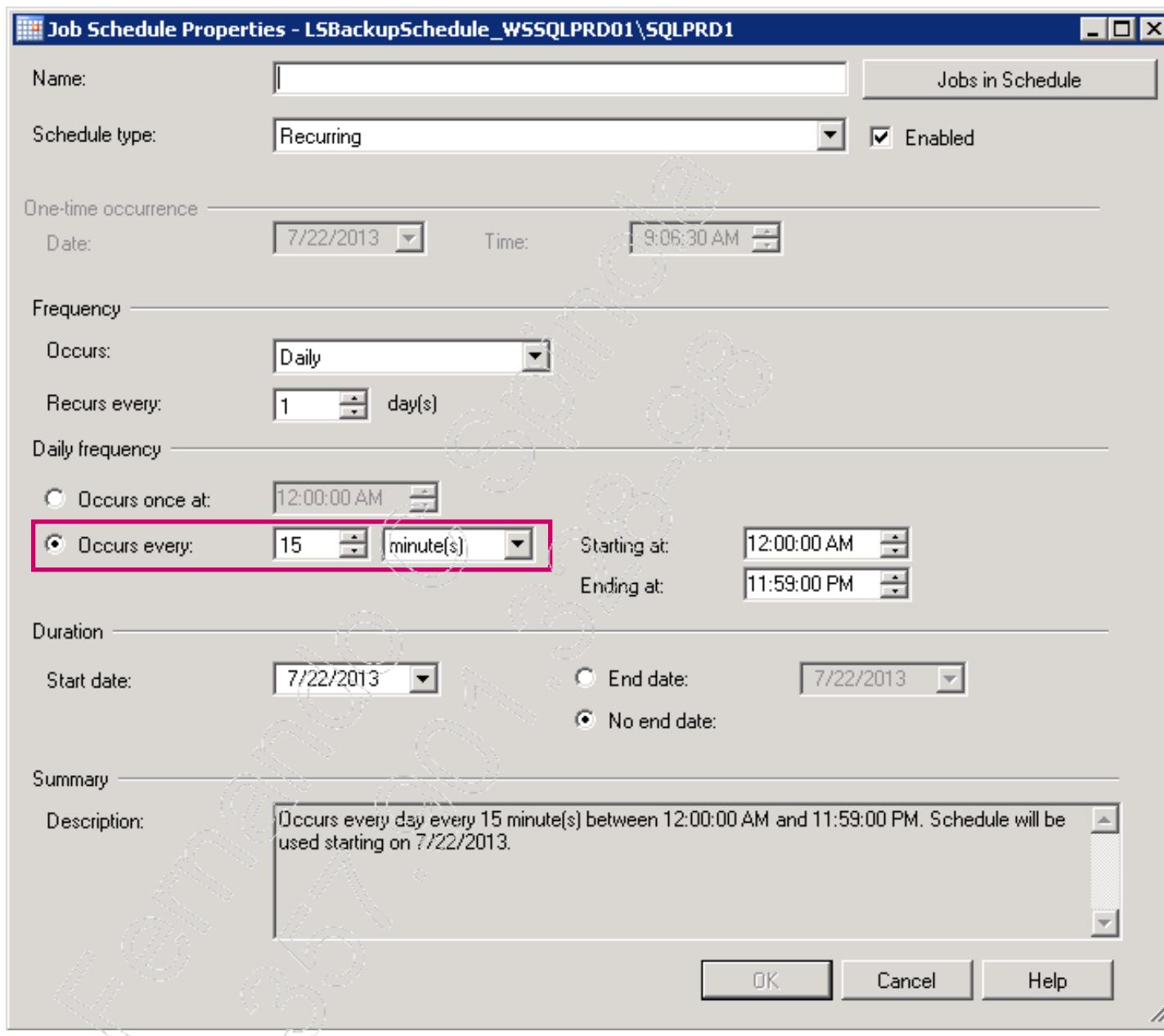


5. Defina o caminho de rede no qual serão gerados os arquivos de backups de log. Para isso, informe o caminho no servidor secundário para envio do arquivo contendo o backup (**Network path to backup folder**) ou então informe o diretório no servidor primário no qual o backup do log deva ser criado (segunda opção);

É importante indicar a opção de exclusão dos arquivos de backup que tiverem mais de n dias (**Delete files older than**). Dessa forma, haverá um controle automático, pois os arquivos são mantidos pelo tempo determinado. No SQL Server Agent, a opção **Backup job** indicará o nome da tarefa que será responsável pela realização dos backups.

6. Selecione o botão **Schedule** para agendar o intervalo para realização dos backups de log. Isso é configurável conforme a necessidade de atualização do ambiente. Para sistemas críticos, recomenda-se o tempo entre 5 e 30 minutos e para sistemas não críticos de 60 a 1440 minutos;

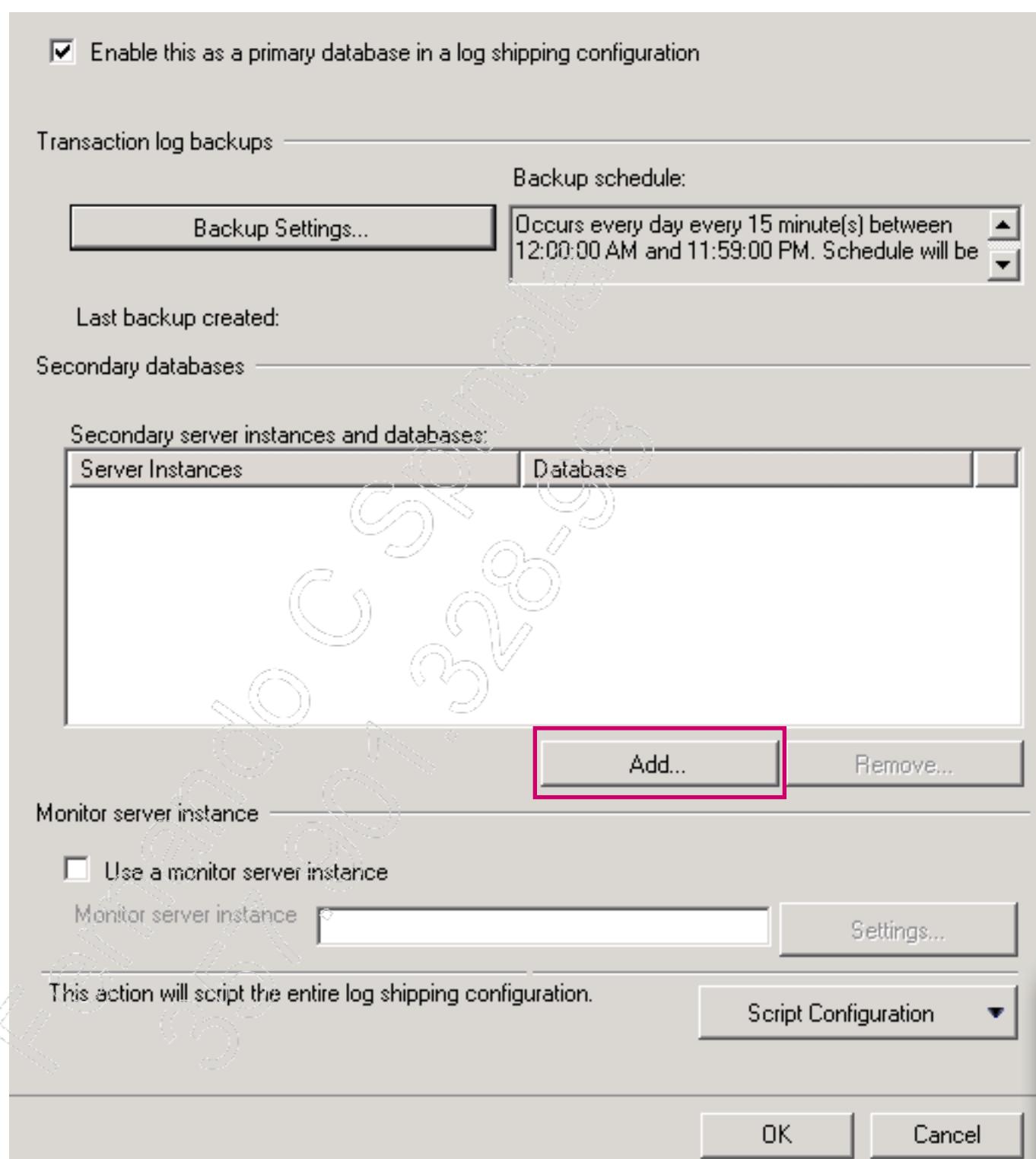
7. Informe o nome da tarefa (**Name**) e indique o tipo de agendamento (**Schedule type**). Neste caso, selecione a opção **Recurring**. Defina a frequência para atualização. Para os casos de **Log Shipping**, selecione a opção **Daily** e indique a periodicidade durante o dia em que o backup irá ocorrer. Selecione, por exemplo, 15 minutos:



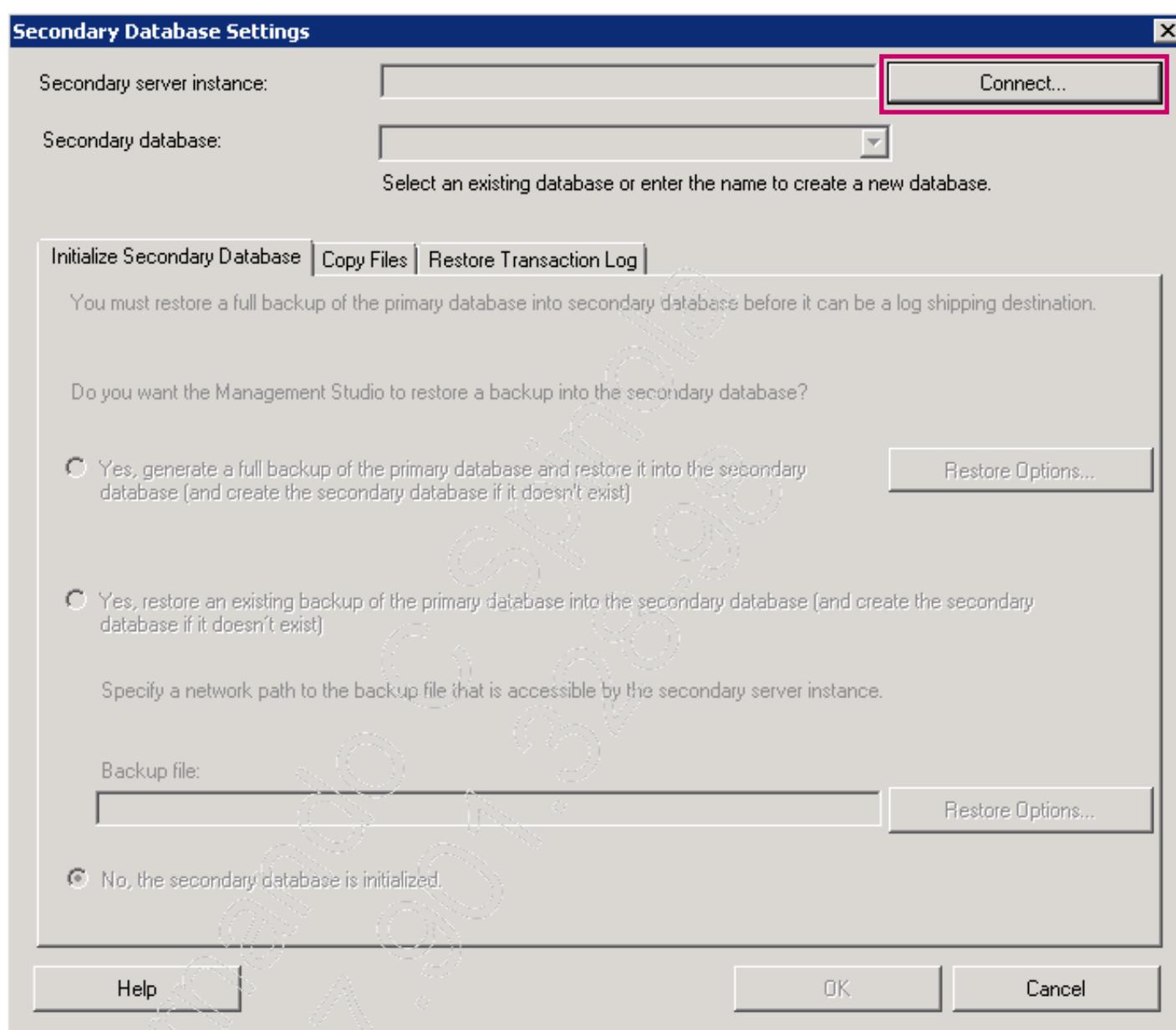
8. Clique no botão **OK** para retornar à tela de configuração e dar continuidade ao processo;

SQL 2014 - Módulo III

9. Vamos agora configurar a instância secundária (**Secondary Database**). Para isso, selecione o botão **Add...**:

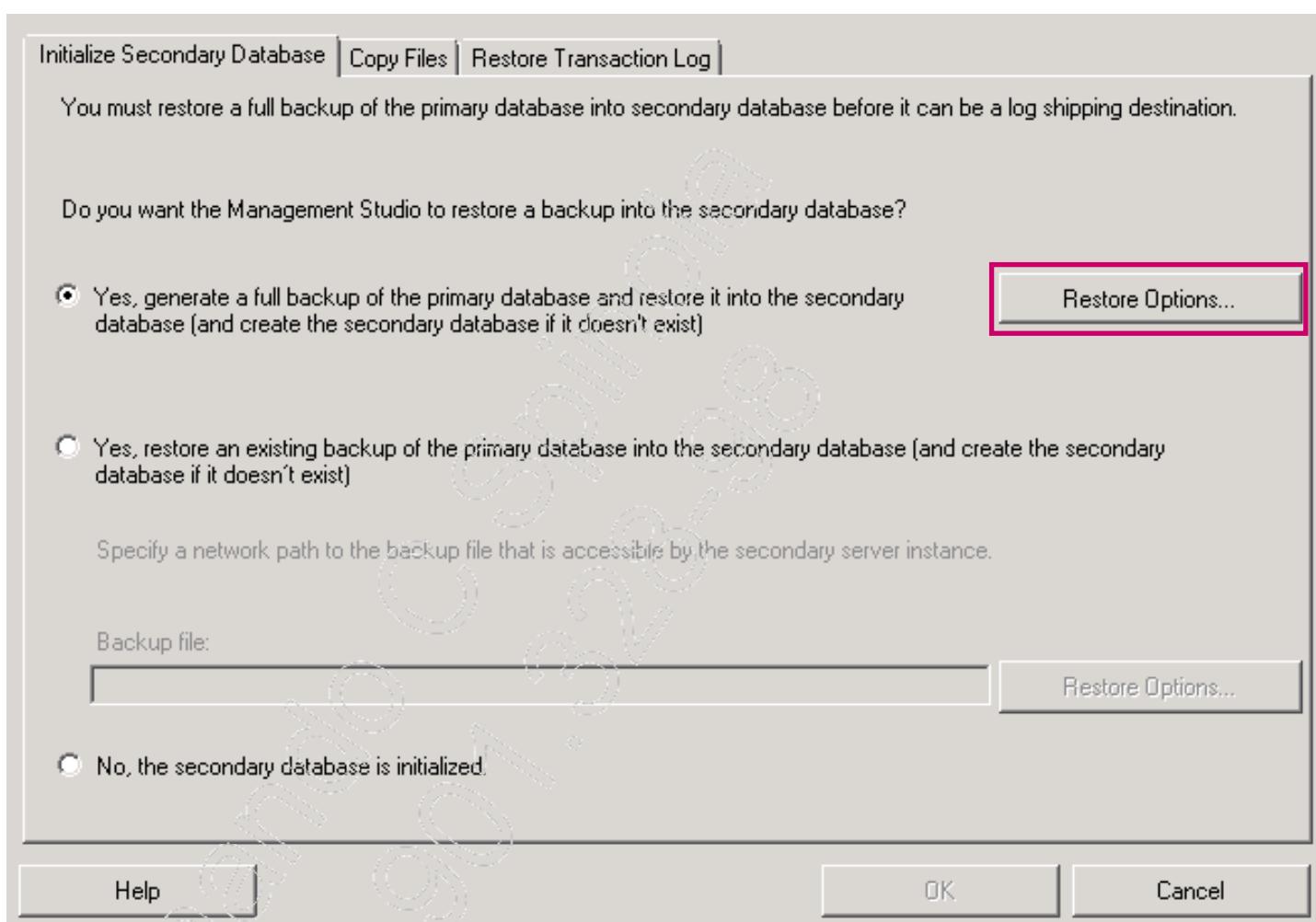


10. Em seguida, conecte ao servidor secundário para configuração do **Log Shipping** por meio do botão **Connect...**:

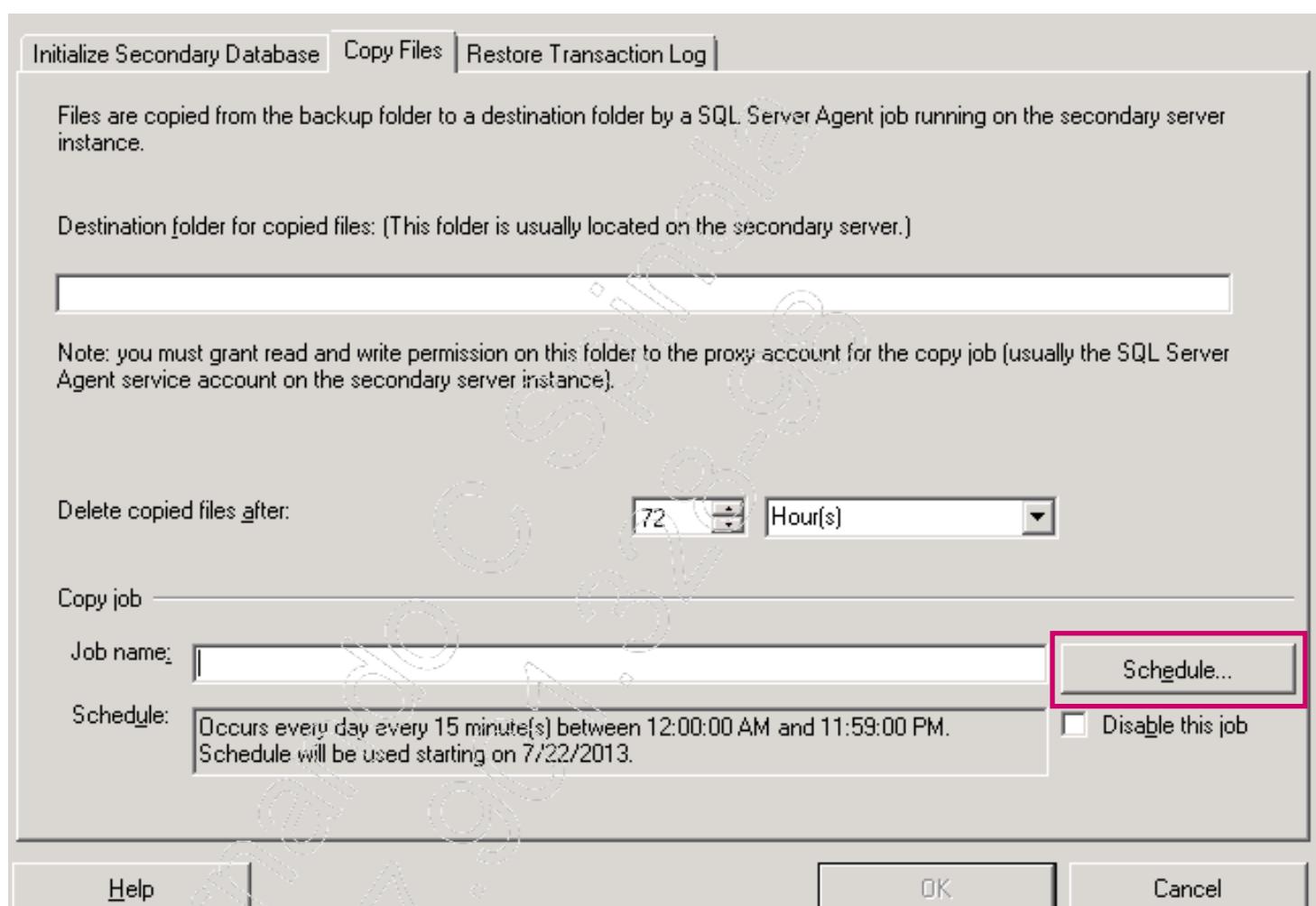


11. Indique o servidor e a instância, além do nome e da senha do usuário para conexão no servidor secundário;

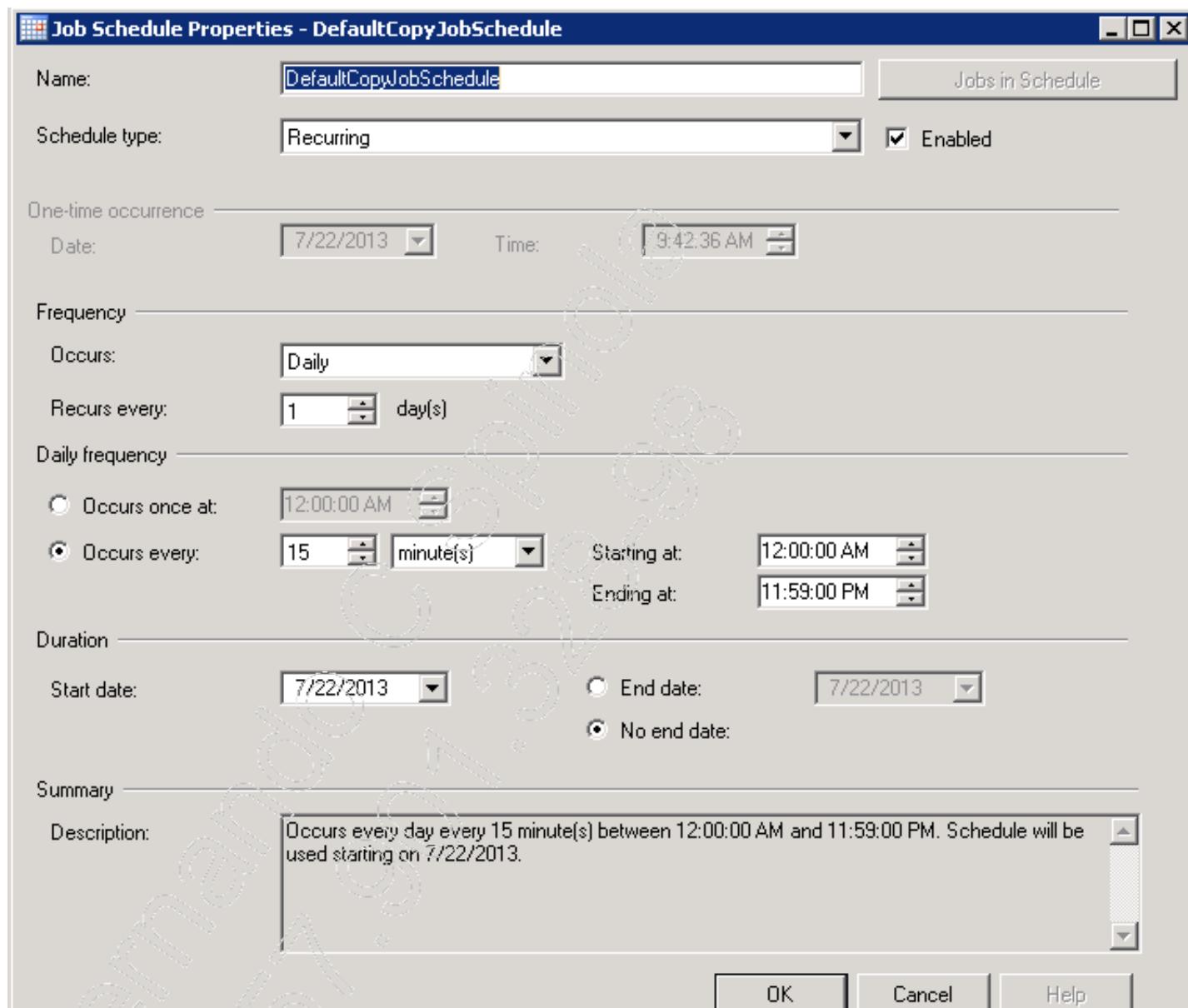
12. Marque a opção **Yes, generate a full backup...** caso queira realizar um backup do banco de dados do servidor principal e restaurá-lo no servidor secundário, ou a segunda opção, se o banco de dados já existir no servidor secundário. Selecione a primeira opção. Em seguida, clique em **Restore Options...**:



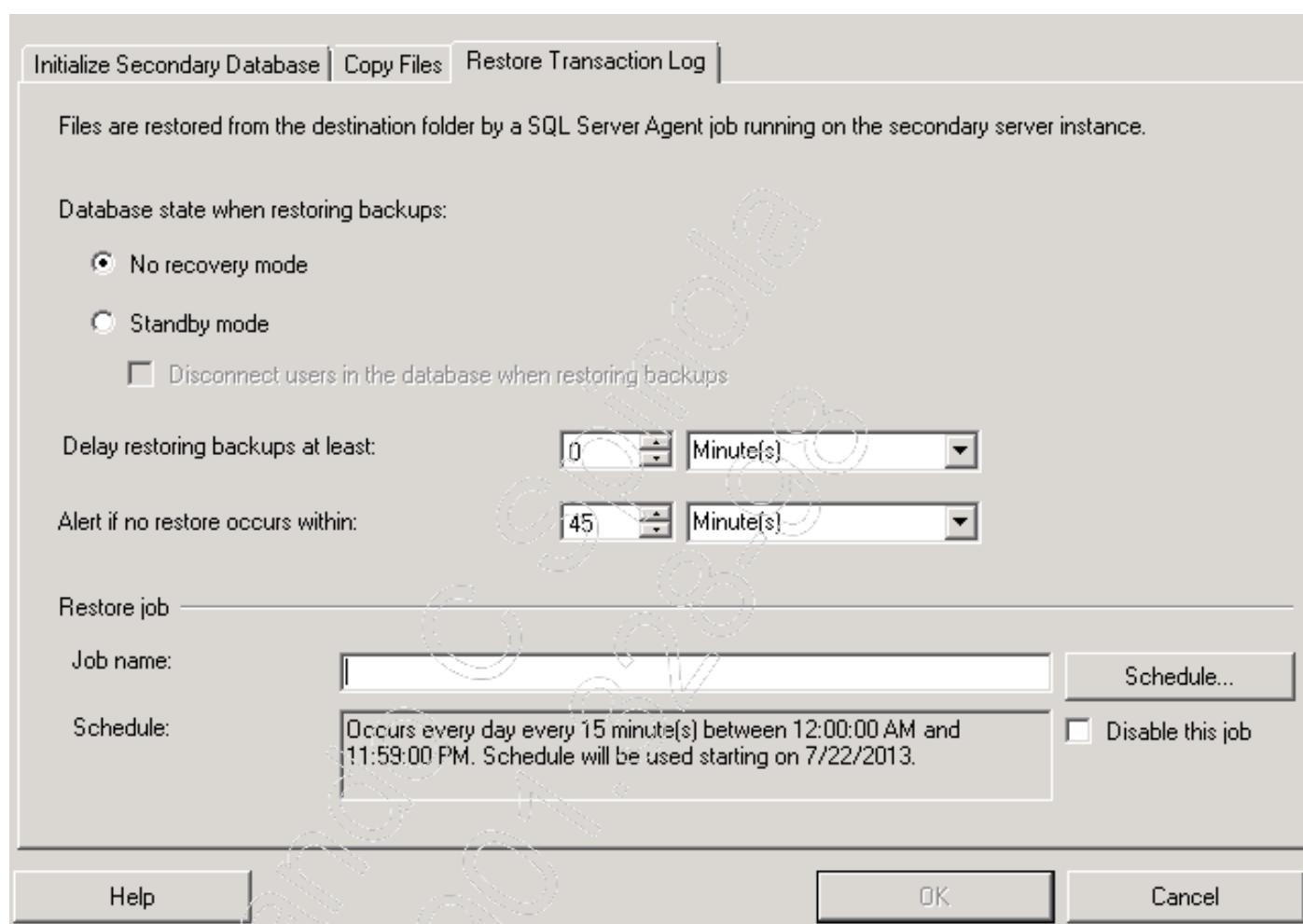
13. Indique o diretório compartilhado no servidor secundário que deverá receber os arquivos de log. Em seguida, defina o tempo para eliminação dos arquivos copiados (**Delete copied files after**). Depois, informe o nome da tarefa (**Job name**) no servidor secundário para aplicação dos backups de log recebidos do servidor secundário. Agora, clique no botão **Schedule...**:



14. Selecione o periodo para cópia dos arquivos de log. É importante que os horários de cópia sejam posteriores aos de geração do backup, por exemplo, cinco minutos depois:



15. Defina que a cópia deve ocorrer diariamente, a cada 10 minutos, iniciando às **12:10:00**. Dessa forma, ela vai acontecer sempre 10 minutos após ser executada a tarefa de backup no servidor principal. Em seguida, clique em **OK**. A última etapa é a definição do procedimento de restauração:



16. Indique o estado do banco de dados após a restauração dos backups (sem modo de recuperação ou no modo standby). Indique o nome da tarefa de restauração (**Job name**). Selecione o botão **Schedule** para agendar a tarefa de restauração. Em seguida, clique em **OK**;

17. Ao retornar à tela inicial, clique em **OK** para configuração do **Log Shipping**.

SQL 2014 - Módulo III

O script a seguir poderá ser executado como alternativa ao processo descrito anteriormente. Nesse caso, é importante que os diretórios e instâncias estejam definidos:

```
DECLARE @LS_BackupJobId AS uniqueidentifier
DECLARE @LS_PrimaryId AS uniqueidentifier
DECLARE @SP_Add_RetCode As int

EXEC @SP_Add_RetCode = master.dbo.sp_add_log_shipping_primary_
database
    @database = N'exemplo_logshipping'
    ,@backup_directory = N'\\AUNOPRINCIPAL\LOG_SHIP'
    ,@backup_share = N'\\AUNOSECUNDARIO\LOG_SHIP'
    ,@backup_job_name = N'LBackup_exemplo_logshipping'
    ,@backup_retention_period = 4320
    ,@backup_compression = 2
    ,@backup_threshold = 60
    ,@threshold_alert_enabled = 1
    ,@history_retention_period = 5760
    ,@backup_job_id = @LS_BackupJobId OUTPUT
    ,@primary_id = @LS_PrimaryId OUTPUT
    ,@overwrite = 1

IF (@@ERROR = 0 AND @SP_Add_RetCode = 0)
BEGIN

DECLARE @LS_BackUpScheduleUID As uniqueidentifier
DECLARE @LS_BackUpScheduleID AS int

EXEC msdb.dbo.sp_add_schedule
    @schedule_name =N'LBackupSchedule_WSSQLPRD01\IMPACTA1'
    ,@enabled = 1
    ,@freq_type = 4
    ,@freq_interval = 1
    ,@freq_subday_type = 4
    ,@freq_subday_interval = 15
    ,@freq_recurrence_factor = 0
    ,@active_start_date = 20130722
    ,@active_end_date = 99991231
    ,@active_start_time = 0
    ,@active_end_time = 235900
```

```
,@schedule_uid = @LS_BackUpScheduleUID OUTPUT  
,@schedule_id = @LS_BackUpScheduleID OUTPUT  
  
EXEC msdb.dbo.sp_attach_schedule  
    @job_id = @LS_BackupJobId  
    ,@schedule_id = @LS_BackUpScheduleID  
  
EXEC msdb.dbo.sp_update_job  
    @job_id = @LS_BackupJobId  
    ,@enabled = 1  
  
END  
  
EXEC master.dbo.sp_add_log_shipping_alert_job  
  
EXEC master.dbo.sp_add_log_shipping_primary_secondary  
    @primary_database = N'exemplo_logshipping'  
    ,@secondary_server = N'10.171.171.154\IMPACTA'  
    ,@secondary_database = N'exemplo_logshipping'  
    ,@overwrite = 1  
  
-- ***** End: Script to be run at Primary: [WSSQLPRD01\IMPACTA]  
*****  
  
-- Execute the following statements at the Secondary to configure  
Log Shipping  
-- for the database [10.171.171.154\IMPACTA].[exemplo_  
logshipping],  
-- the script needs to be run at the Secondary in the context of  
the [msdb] database.  
-----  
-----  
-- Adding the Log Shipping configuration  
  
-- ***** Begin: Script to be run at Secondary: [10.171.171.154\  
IMPACTA] *****  
  
DECLARE @LS_Secondary_CopyJobId AS uniqueidentifier  
DECLARE @LS_Secondary_RestoreJobId AS uniqueidentifier  
DECLARE @LS_Secondary_SecondaryId AS uniqueidentifier
```

SQL 2014 - Módulo III

```
DECLARE @LS_Add_RetCode As int

EXEC @LS_Add_RetCode = master.dbo.sp_add_log_shipping_secondary_
primary
    @primary_server = N'WSSQLPRD01\IMPACTA'
    ,@primary_database = N'exemplo_logshipping'
    ,@backup_source_directory = N'\\AUNOSECUNDARIO\LOG_SHIP'
    ,@backup_destination_directory = N'\\AUNOSECUNDARIO\log_
ship'
    ,@copy_job_name = N'LSCopy_WSSQLPRD01\IMPACTA_exemplo_
logshipping'
    ,@restore_job_name = N'job_restore_exemplo_logshipping'
    ,@file_retention_period = 4320
    ,@overwrite = 1
    ,@copy_job_id = @LS_Secondary_CopyJobId OUTPUT
    ,@restore_job_id = @LS_Secondary_RestoreJobId OUTPUT
    ,@secondary_id = @LS_Secondary_SecondaryId OUTPUT

IF (@@ERROR = 0 AND @LS_Add_RetCode = 0)
BEGIN

DECLARE @LS_SecondaryCopyJobScheduleUID As uniqueidentifier
DECLARE @LS_SecondaryCopyJobScheduleID AS int

EXEC msdb.dbo.sp_add_schedule
    @schedule_name =N'DefaultCopyJobSchedule'
    ,@enabled = 1
    ,@freq_type = 4
    ,@freq_interval = 1
    ,@freq_subday_type = 4
    ,@freq_subday_interval = 15
    ,@freq_recurrence_factor = 0
    ,@active_start_date = 20130722
    ,@active_end_date = 99991231
    ,@active_start_time = 1000
    ,@active_end_time = 235900
    ,@schedule_uid = @LS_SecondaryCopyJobScheduleUID OUTPUT
    ,@schedule_id = @LS_SecondaryCopyJobScheduleID OUTPUT

EXEC msdb.dbo.sp_attach_schedule
    @job_id = @LS_Secondary_CopyJobId
    ,@schedule_id = @LS_SecondaryCopyJobScheduleID
```

```
DECLARE @LS_SecondaryRestoreJobScheduleUID As uniqueidentifier
DECLARE @LS_SecondaryRestoreJobScheduleID AS int

EXEC msdb.dbo.sp_add_schedule
    @schedule_name =N'DefaultRestoreJobSchedule',
    @enabled = 1
    ,@freq_type = 4
    ,@freq_interval = 1
    ,@freq_subday_type = 4
    ,@freq_subday_interval = 15
    ,@freq_recurrence_factor = 0
    ,@active_start_date = 20130722
    ,@active_end_date = 99991231
    ,@active_start_time = 1500
    ,@active_end_time = 235900
    ,@schedule_uid = @LS_SecondaryRestoreJobScheduleUID OUTPUT
    ,@schedule_id = @LS_SecondaryRestoreJobScheduleID OUTPUT

EXEC msdb.dbo.sp_attach_schedule
    @job_id = @LS_Secondary_RestoreJobId
    ,@schedule_id = @LS_SecondaryRestoreJobScheduleID

END

DECLARE @LS_Add_RetCode2 As int

IF (@@ERROR = 0 AND @LS_Add_RetCode = 0)
BEGIN

EXEC @LS_Add_RetCode2 = master.dbo.sp_add_log_shipping_secondary_
database
    @secondary_database = N'exemplo_logshipping'
    ,@primary_server = N'WSSQLPRD01\IMPACTA'
    ,@primary_database = N'exemplo_logshipping'
    ,@restore_delay = 0
    ,@restore_mode = 0
    ,@disconnect_users = 0
    ,@restore_threshold = 45
    ,@threshold_alert_enabled = 1
```

```
    ,@history_retention_period      = 5760
    ,@overwrite = 1

END

IF (@@error = 0 AND @LS_Add_RetCode = 0)
BEGIN

EXEC msdb.dbo.sp_update_job
    @job_id = @LS_Secondary_CopyJobId
    ,@enabled = 1

EXEC msdb.dbo.sp_update_job
    @job_id = @LS_Secondary_RestoreJobId
    ,@enabled = 1

END
```

Este procedimento fará com que o **Log Shipping** seja configurado e fique ativado tanto para o servidor primário quanto para o secundário.

12.3. Database Mirroring

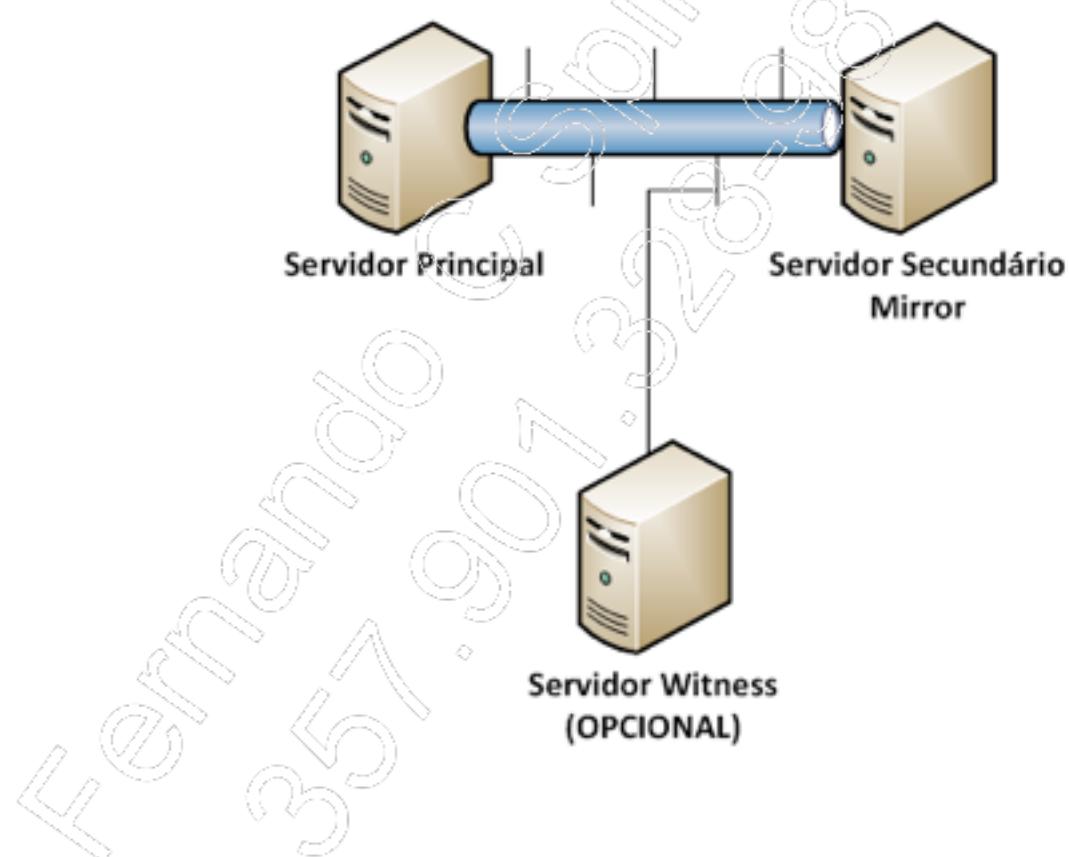
O recurso **Database Mirroring** foi introduzido na versão 2005 e tem como objetivo promover a alta disponibilidade em cenários que envolvam recuperação de desastres (Disaster Recovery). Este recurso fornece um mecanismo de sincronização dos dados entre os bancos de dados, seja de forma síncrona ou assíncrona. Além disso, oferece baixa latência dos dados, já que eles são sincronizados entre os bancos de forma automática.

Podemos configurar um **Database Mirroring** com dois servidores, cada um com uma instância criada (com mesmo nome ou não) e que possam ter conexão entre si. Opcionalmente, é possível ter um terceiro servidor chamado de testemunha (witness), que pode assumir o papel de verificar o andamento das atividades no servidor principal e no secundário. Ao notar que o servidor principal não responde aos chamados, o witness promove o servidor secundário como servidor principal. Esse processo é chamado de **Failover Automático**. Caso não utilizemos o componente witness, o processo de failover passa a ser totalmente manual.

O database principal deve ser configurado no modelo de recuperação **Full** (**Recovery Model Full**), pois a configuração envia os logs de um servidor para o outro. Um banco de dados configurado como **Bulk_Logged** não poderá enviar os logs para o database mirror.

É necessário criar o banco de dados do espelhamento com uma operação de **Restore** (com a cláusula **Norecovery**) feita de um backup do banco de dados principal, seguida por Restores sequenciais do transaction log desse mesmo banco de dados principal.

O database mirror deve ter o mesmo nome do database principal. Como o database mirror ficará em estado de recuperação (**Load**), ele não poderá ser acessado. Podemos criar um database snapshot no database mirror para poder ler seus dados indiretamente.



Existem três tipos de proteção disponíveis para o **Database Mirroring**:

- **Máxima Segurança (High Safety)**: A replicação dos dados neste modo ocorre de maneira síncrona entre os servidores. Dessa forma, as operações serão completadas somente se ocorrer **commit** nos dois servidores de forma simultânea. Isso pode implicar uma significativa perda de performance, já que depende de os participantes informarem que efetivaram as operações em cada banco de dados da configuração. Ao utilizarmos a configuração com witness, este é o modelo adotado;
- **Alta Performance (High Performance)**: Neste modo, a replicação ocorre de maneira assíncrona e o processo de gravação é realizado obedecendo ao protocolo **Two-Phase-Commit**, no qual os dados são gravados primeiramente no servidor principal e depois no servidor mirror. Isso resulta em um tempo melhor de resposta. Como não se pode verificar se a transação foi efetivada no servidor mirror, essa opção oferece uma performance melhor em relação à máxima segurança, porém, com menor grau de segurança. Ao optarmos por esta opção, não poderemos utilizar uma instância witness para gerenciamento;
- **Alta Proteção (High Protection)**: Neste modo, a proteção do mecanismo de transferência é síncrono e não utiliza witness server. É uma proteção intermediária entre as duas anteriores.

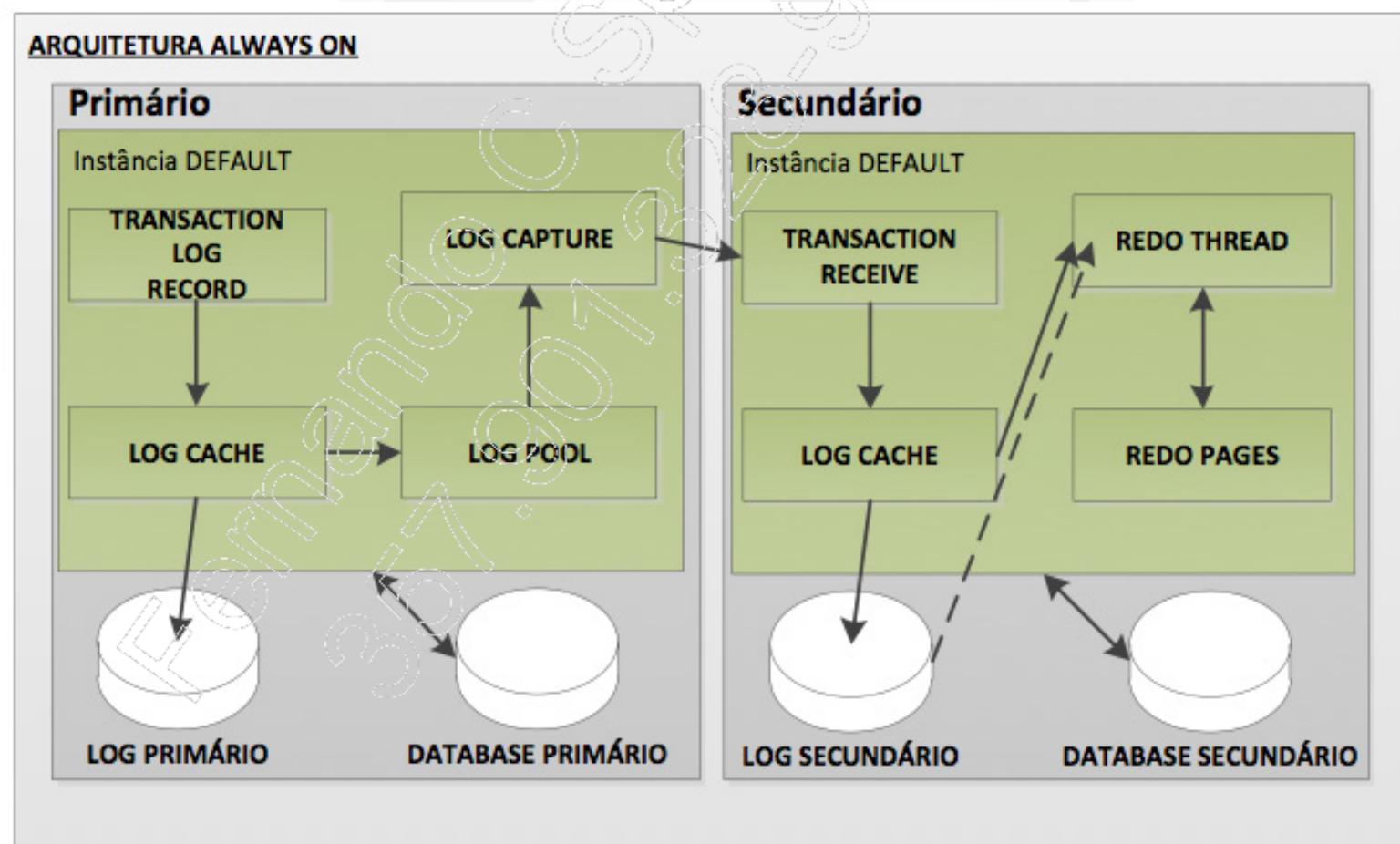
Este recurso entrou em desuso a partir da versão SQL Server 2012 e, como alternativa, tornou-se recomendável o uso do recurso **Always ON**.

12.4. Always ON – SQL 2014

A arquitetura de alta disponibilidade do SQL Server ganhou um novo componente capaz de garantir a disponibilidade de ambientes críticos e integrados em cloud. Por meio do **Always ON**, é possível manter até quatro réplicas para cada servidor primário. Semelhante ao **Database Mirroring**, ele não apenas o estende como também oferece ganhos de performance com relação a dois itens:

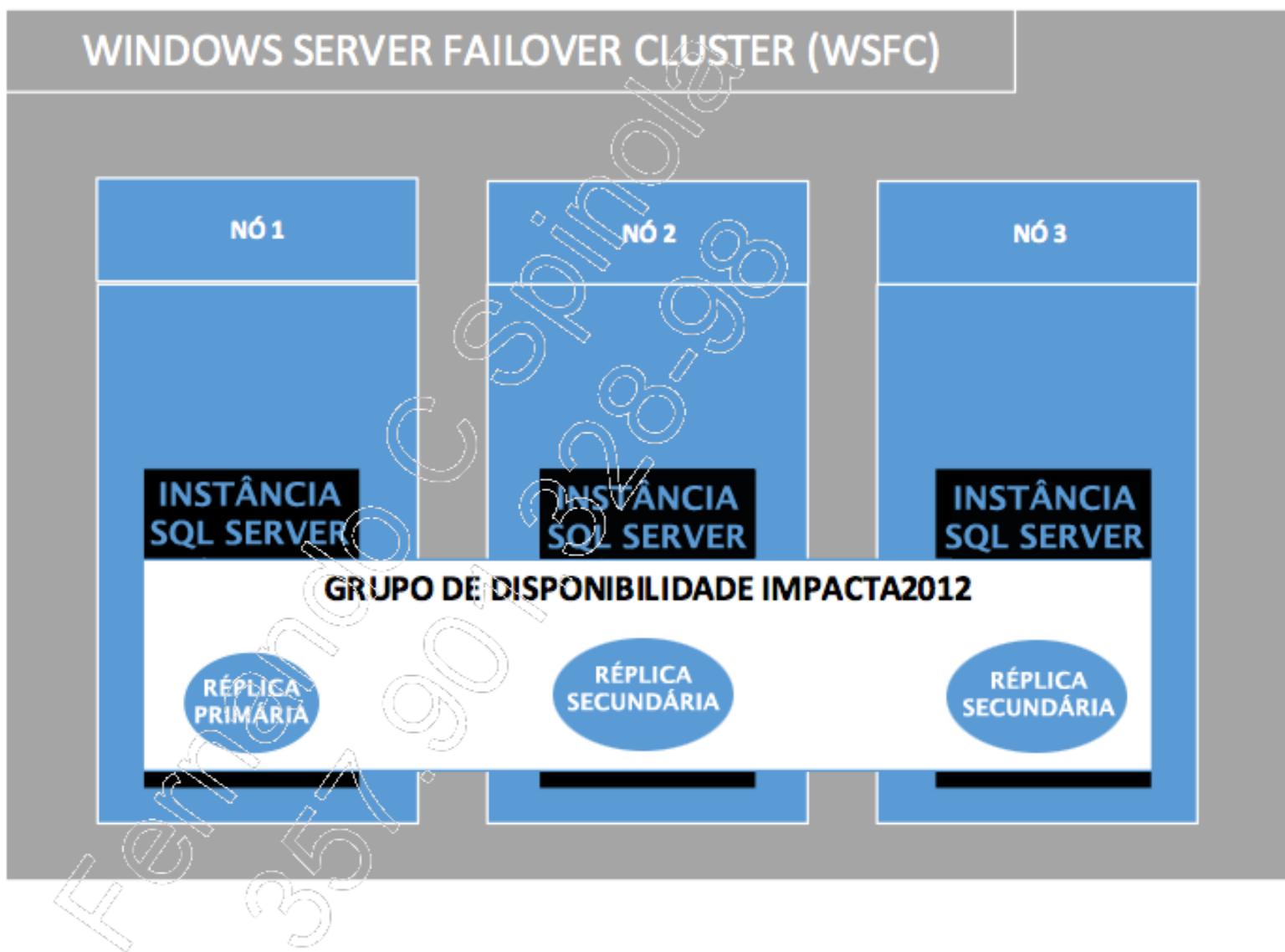
- Eficiência de leitura e gravação (I/O) de páginas de dados no servidor secundário;
- Melhoria na leitura e gravação através de otimizações realizadas no arquivo de log usando o recurso de log pool.

A seguir, ilustramos o processo de captura e envio de log do **Always ON**:



12.4.1. Arquitetura Windows Server Failover Cluster e Always ON SQL Server 2014

Como dissemos anteriormente, o **Always ON** permite controlar uma réplica primária e até quatro réplicas secundárias. Na imagem a seguir, ilustramos uma réplica primária e duas secundárias. Com isso, é criado um grupo de disponibilidade que permite que, em caso de falha de um nó, os demais nós possam assumir o papel e prosseguir com as atividades realizadas pelo nó que continha a réplica primária.



Para implementação do **Always ON**, há a necessidade de utilização do Windows Server Failover Clustering, conforme indicado na imagem anterior. Este recurso permite a alta disponibilidade em casos de desastres em servidores com recursos de alta importância para a continuidade de negócios, como o Microsoft Exchange e o Microsoft SQL Server. Em caso de falha, a realocação dos serviços relativos àquele nó para outro nó pode ser feita de forma manual ou automática através do procedimento denominado failover.

Os nós do WSFC possuem os seguintes recursos:

- **Monitor de integridade:** Este recurso monitora a integridade dos nós e do nó primário através da análise chamada de heartbeat. Ao detectar alguma falha, a integridade do cluster pode ser colocada em questionamento pelo monitor de integridade, pois cada nó deposita um voto no quórum do cluster. Ao encontrar uma falha, o voto não é depositado e, com isso, detecta-se a falha no nó;
- **Gerenciamento de recursos:** Os recursos individuais dos nós, como armazenamento em disco compartilhado e interface de rede, são gerenciados através deste recurso;
- **Gerenciamento de metadados:** Os metadados de aplicativos são mantidos em cada nó do cluster, assim como o serviço do WSFC. Essas informações são copiadas automaticamente entre os nós do cluster;
- **Coordenação do failover:** A política estabelecida em caso de falha de um dos nós determina a transferência automática ou manual. Com isso, quando um failover ocorre, os nós e os aplicativos são notificados para que possam reagir de maneira adequada ao evento.

O serviço de clusterização do WSFC pode verificar e definir a necessidade de reiniciar uma instância SQL Server ou até mesmo de realizar um failover em outro nó do mesmo cluster WSFC.

Para o SQL Server, a plataforma do WSFC oferece a possibilidade de registrar componentes como recursos do cluster WSFC. Cada nó que compõe esse cluster é chamado de Failover Cluster Instance (FCI). A FCI depende do uso de recurso de compartilhamento de disco (via Fibre Channel ou via SAN) e de uma rede virtual, contendo IPs Virtuais em sub-redes diferentes.

Ao ocorrer um failover, o serviço WSFC realiza a transferência de todos os recursos de instância definidos para outro nó designado. Após realizar essa operação, a instância SQL Server é reiniciada no novo nó e os bancos de dados são recuperados. A FCI somente pode estar ativa em um único nó ao mesmo tempo.

Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes do capítulo.

- **Log Shipping** é o recurso de recuperação de desastre, que utiliza como base o backup regular de log e envio deste ao servidor secundário, que recebe e aplica esse backup de modo contínuo;
- **Database Mirroring** é um recurso que visa a promover a alta disponibilidade em cenários que envolvam recuperação de desastres (Disaster Recovery). Ele existe desde a versão 2005 do SQL Server, mas entrou em desuso a partir da versão 2014;
- A alta disponibilidade em bancos de dados SQL Server pode ser implementada através do recurso chamado de **Always ON**. Este recurso substituiu o **Database Mirroring**;
- Por meio do **Always ON**, é possível manter até quatro réplicas para cada servidor primário.

12

**Alta disponibilidade
Teste seus conhecimentos**

Fernando Spinola
357.907.0008



IMPACTA
EDITORA

1. Quais são os tipos de proteção disponíveis no Database Mirroring?

- a) Alta disponibilidade e máxima segurança.
- b) Alta disponibilidade e alta performance.
- c) MS Cluster
- d) Log Shipping
- e) Alta disponibilidade, máxima segurança e alta performance.

2. Qual recurso deve ser utilizado no lugar do Database Mirroring a partir do SQL Server 2012?

- a) Always ON
- b) Backup
- c) MS Cluster
- d) Log Shipping
- e) Database Mirroring 2014

3. Quantos servidores são necessários para implementarmos um Database Mirroring?

- a) 1
- b) 2
- c) 3
- d) 2 ou 3
- e) 4

4. Qual das alternativas a seguir não descreve uma diferença entre Log Shipping e replicação de dados?

- a) Não é obrigatória a utilização de chaves primárias no Log Shipping.
- b) Replicação de dados é um recurso que pode enviar tabelas individuais.
- c) Log Shipping é recomendado para DR (Disaster Recover).
- d) Um servidor secundário de Log Shipping pode ser usado como principal.
- e) Replicação e Log Shipping são recursos utilizados para DR (Disaster Recover).

5. Qual(is) modo(s) de recuperação (Recovery Mode) deve(m) ser utilizado(s) para configuração do Log Shipping?

- a) Full
- b) Full ou Bulk Logged.
- c) Simple
- d) Full, Bulk Logged e Simple.
- e) Full e Simple.

12

Alta disponibilidade Mãos à obra!

Fernando Spinola
357.907.3298



IMPACTA
EDITORA

Laboratório 1

Neste laboratório, vamos configurar o espelhamento em um database. Para tanto, vamos utilizar dois servidores: um será o servidor **Principal** e o outro será o servidor **Mirror** (ou **Espelho**).

Antes de começar as configurações, preencha o quadro a seguir:

Nome do servidor escolhido para rodar o database **Principal**: _____

Nome do servidor escolhido para rodar o database **Mirror**: _____

A sequência para realizar este exercício é a seguinte:

- Configurar parâmetros de Startup do serviço **MSSQLServer**;
- Preparar o ambiente dos dados para o espelhamento no servidor **Principal**;
- Preparar o ambiente dos dados para o espelhamento no servidor de espelhamento;
- Configurar o espelhamento;
- Executar comandos DML para testar o espelhamento;
- Mudar os papéis dos servidores;
- Executar comandos DML para testar o espelhamento;
- Desfazer o espelhamento.

A - Configurando parâmetros de Startup do serviço MSSQLServer

Para que a configuração de espelhamento possa funcionar, devemos, primeiro, fazer a configuração a seguir. Execute estes passos nos dois servidores que farão parte da configuração de espelhamento:

1. Clique no botão **Start** e abra a opção **All Programs**. Em seguida, escolha **Microsoft SQL Server 2014, Configuration Tools e SQL Server Configuration Manager**;
2. Na tela que será exibida, do lado direito, clique com o botão direito do mouse sobre o serviço **SQL Server (MSSQLServer)** e escolha a opção **Properties**;
3. Na tela seguinte, clique na guia **Advanced**;
4. Posicione o mouse no final da opção de configuração **Startup Parameters** e escreva exatamente assim: **-T1400** (ponto e vírgula e **-T1400**);
5. Desligue e, em seguida, ligue novamente o serviço **SQL Server (MSSQLServer)**.

B - Preparando o ambiente dos dados para o espelhamento no servidor principal

 Esta etapa deve ser feita no servidor que vai possuir o database **Principal**.

1. Abra o **SQL Server Management Studio** e conecte-se ao servidor que terá o database **Principal** (origem dos dados a serem espelhados) com a autenticação do Windows;
2. Abra o **Script_01** da pasta **Capitulo_12** com a autenticação do Windows no servidor que será a origem dos dados do espelhamento. Pressione F5 para executá-lo.

C – Preparando o ambiente dos dados para o espelhamento no servidor de espelhamento

1. No Windows Explorer, crie uma pasta chamada Backup na raiz do disco C:\ (C:\Backup);
2. Compartilhe esta pasta para Everyone **Full Control**:
 - 2.1. Clique com o botão direito do mouse sobre a pasta **Backup**;
 - 2.2. Escolha a opção **Sharing and Security**;
 - 2.3. Clique na guia **Sharing**;
 - 2.4. Selecione a opção **Share this folder**;
 - 2.5. Clique no botão **Permissions**;
 - 2.6. Clique na opção **Full Control**.
3. Copie o arquivo de backup criado pelo **Script_01** para o servidor que vai conter o database de espelhamento:
 - 3.1. No **Windows Explorer**, expanda a pasta C:\Backup;
 - 3.2. Clique com o botão direito do mouse sobre o arquivo **Backup_Ginasio.Bak** e escolha a opção **Copy**;
 - 3.3. Expand o ícone **My Network Places**;
 - 3.4. Expand **Entire Network**;
 - 3.5. Expand **Microsoft Windows Network**;
 - 3.6. Expand o **Domínio** da sala de aula;
 - 3.7. Expand o servidor que será utilizado como espelhamento;
 - 3.8. Clique sobre a pasta **Backup** e note que ela está vazia;

3.9. Do lado direito da tela, clique com o botão direito do mouse e escolha a opção **Past**;

3.10. Note que o arquivo **Backup_Ginasio.Bak** agora existe na pasta **C:\Backup** do servidor que será utilizado como espelhamento do database **Ginasio**.

4. Abra o **SQL Server Management Studio** conectando-se com a autenticação do Windows;

5. Abra o **Script_02** da pasta **Capítulo_12** e pressione F5 para executá-lo.

D - Configurando o espelhamento



! Esta etapa deverá ser feita no servidor **Principal**.

1. No **SQL Server Management Studio** do servidor **Principal**, expanda a pasta **Databases**, clique com o botão direito do mouse sobre o database **Ginasio**, escolha a opção **Tasks** e, em seguida, escolha a opção **Mirror...**;

2. Na tela que será exibida, clique no botão **Configure Security**;

3. Na tela seguinte, clique no botão **Next**;

4. Na próxima tela, escolha a opção **No** e, em seguida, clique em **Next**;

5. Escolha a opção **Mirror server instance** e clique em **Next** na tela seguinte;

6. A próxima tela deverá ter as seguintes configurações:

- No campo **Principal Server Instance**, deve estar marcado o nome do servidor **Principal**;
- No campo **Listener Port**, deve estar marcado **5022**;

- No campo **Endpoint Name**, escreva: **Mirroring_Principal_Ginasio** e, em seguida, clique em **Next**;
- Na próxima tela, clique no botão **Connect**;
- Na tela que será exibida, no campo **Server name**, escolha o nome do servidor que terá o database de espelhamento e conecte-se com a autenticação do Windows;
- No campo **Endpoint name**, escreva **Mirroring_Espelho_Ginasio** e, em seguida, clique em **Next**;
- Não escreva nada na próxima tela e clique em **Next**;
- Observe a lista das configurações que serão executadas e clique em **Finish**;
- Observe o SQL Server executando as tarefas e, assim que terminar com sucesso, clique em **Close**;
- Deverá surgir uma mensagem dizendo que, para iniciar o processo de espelhamento, é necessário clicar no botão **Start Mirroring**. Então, nesta mensagem, clique em **OK** e, em seguida, clique em **Start Mirroring**;
- Clique no botão **OK**.

E – Executando comandos DML para testar o espelhamento

 Esta etapa deverá ser feita no servidor **Principal**.

1. No **SQL Server Management Studio** do servidor **Principal**, abra o **Script_03** da pasta **Capitulo_12** e pressione F5 para executá-lo.

F – Mudando os papéis dos servidores



Esta etapa deverá ser feita no servidor **Principal**.

1. No **SQL Server Management Studio**, expanda a pasta **Databases** e clique com o botão direito do mouse sobre o database **Ginasio**. Escolha a opção **Tasks** e, em seguida, escolha **Mirror...**;
2. Na tela que será exibida, clique no botão **Failover**;
3. Aparecerá uma mensagem dizendo que os servidores vão trocar de papéis. Clique em **Yes**;
4. Clique com o botão direito do mouse sobre a pasta **Databases** e escolha a opção **Refresh**. Note que agora o database **Ginasio** é o database **Mirror**.

G – Executando comandos DML para testar o espelhamento



Esta etapa deverá ser feita no servidor que era o espelho, mas que agora é o **Principal**.

1. No **SQL Server Management Studio** do servidor que tinha o espelhamento, mas que agora é o servidor **Principal**, clique com o botão direito do mouse sobre a pasta **Databases** e escolha a opção **Refresh**;



Note que agora o database **Ginasio** é o **Principal**.

2. Neste servidor, abra o **Script_04** da pasta **Capitulo_12** e pressione F5 para executá-lo.

H – Desfazendo o espelhamento



Esta etapa deverá ser feita no servidor que era o espelho, mas que agora é o **Principal**.

1. No **SQL Server Management Studio** do servidor que tinha o espelhamento, mas que agora é o servidor **Principal**, clique com o botão direito do mouse sobre a pasta **Databases** e escolha a opção **Tasks**. Em seguida, escolha **Mirror**;
2. Na tela que será exibida, clique no botão **Stop Mirroring** e, na mensagem que será exibida, clique em **Yes**;
3. Clique no botão **OK**;
4. Do lado direito da tela, no **Object Explorer**, clique com o botão direito do mouse sobre a pasta **Databases** e escolha a opção **Refresh**;
5. Expanda o database **Ginasio** e a pasta **Tables**;
6. Clique com o botão direito do mouse sobre a tabela **Esportes** e escolha a opção **Open Table**;
7. Note que nesta tabela há vinte registros;
8. Feche todos os scripts abertos nos dois servidores.

Laboratório 2

Neste laboratório, vamos utilizar dois servidores. Um deverá ser o servidor **Primário** (aquele que será a origem dos dados) e o outro deverá ser o servidor **Secundário**, também chamado de **StandBy Server** (aquele que receberá os dados do primário).

Antes de começar as configurações, preencha o quadro a seguir:

Nome do servidor escolhido para ser o **Primário**: _____

Nome do servidor escolhido para ser o **Secundário**: _____

Verifique agora quais serão os passos que precisam ser seguidos para a realização deste exercício:

- Verificar se as contas de usuário a serem utilizadas estão registradas no grupo local **Administrators**;
- Preparar o ambiente no servidor **Primário**:
 - Criação e compartilhamento de uma pasta que vai armazenar os arquivos de backup;
 - Criação de um database com dados para testes;
 - Verificação do serviço SQL Server Agent.
- Preparar o ambiente no servidor **Secundário**:
 - Criação e compartilhamento de uma pasta que vai armazenar os backups recebidos do servidor primário.
- Verificar o serviço SQL Server Agent;
- Configurar o Log Shipping no servidor Primário;

SQL 2014 - Módulo III

- Verificar os resultados;
- Executar comandos DML no database do servidor **Priamário** para testes;
- Verificar os resultados;
- Fazer o database **Standby** deixar de ser **Secundário**.

A – Verificando se as contas de usuário a serem utilizadas estão registradas no grupo local Administrators



Os passos deste item devem ser executados nos dois servidores.

1. No **Desktop**, clique com o botão direito do mouse sobre o ícone **My Computer** e escolha a opção **Manage**;
2. Na tela que será exibida, expanda **Local Users and Groups**, clique sobre a pasta **Groups** e, do lado direito da tela, aplique um duplo-clique no grupo **Administrators**;
3. Neste grupo, deverá haver as seguintes contas de usuário:
 - **Domínio\Alunox** (conta do aluno local);
 - **Domínio\Alunoy** (Conta do aluno que executa os exercícios em grupo);
 - **Domínio\Domain Admins**;
 - **Domínio\ServiçoSQL**;
 - **Administrator** (local).

Se todas essas contas não estiverem no grupo local **Administrators**, é preciso registrá-las. Para isso, siga este procedimento:

- 3.1. Clique no botão **Add**;

- 3.2. Na tela que será exibida, no campo **Look In**, escolha o nome do domínio da sala de aula;
- 3.3. Selecione as contas citadas no item 3 anterior e clique no botão **OK**;
- 3.4. Em seguida, clique em **Apply** e em **OK** novamente;
- 3.5. Feche a tela do **Computer Manager**.

B – Preparando o ambiente no servidor primário: criação e compartilhamento de uma pasta que vai armazenar os arquivos de backup

1. Conecte-se ao Windows com a conta do **Administrator** do domínio (cuja senha é a palavra **password** com todas as letras escritas em minúsculo);
2. No **Windows Explorer**, na raiz do disco **C:**, crie uma pasta chamada **Logs_A_Eviar**;
3. Compartilhe esta pasta da seguinte maneira:
 - 3.1. Clique com o botão direito do mouse sobre a pasta **C:\Logs_A_Eviar** e escolha a opção **Sharing and Security**;
 - 3.2. Clique na guia **Sharing**;
 - 3.3. Clique no botão **Permission**;
 - 3.4. Clique no botão **Add**;
 - 3.5. Na tela que será exibida, no campo **Enter the object name to select (examples)**, escreva **ServicoSQL** e clique no botão **Check Names**. Neste campo, deverá aparecer a seguinte indicação: **servicosql s. servicosql (servicosql@dom1.com)**. Clique no botão **OK**;
 - 3.6. Note que, no campo **From this location**, deverá aparecer o nome do domínio da sala de aula;
 - 3.7. Clique na opção **Full Control**;

- 3.8. Clique novamente no botão **Add** e, na tela que será exibida, no campo **Enter the object name to select (examples)**, escreva **Administrator** (no singular). Em seguida, clique no botão **Check Names**. Neste campo, deverá aparecer a palavra **Administrator** (sublinhada). Clique no botão **OK**;
- 3.9. Note que, no campo **From this location**, deverá aparecer o nome do domínio da sala de aula;
- 3.10. Clique na opção **Full Control**;
- 3.11. Pressione novamente o botão **Add** e, na tela que será exibida, clique no botão **From Location**. Clique sobre o nome do servidor local e no botão **OK**;
- 3.12. No campo **Enter the object name to select (examples)**, escreva **Administrators** (no plural) e clique no botão **Check Names**. Neste campo, deverá aparecer a palavra **Administrators** (sublinhada). Clique no botão **OK**;
- 3.13. Note que, no campo **From this location**, deverá aparecer a seguinte frase: **NomeDoComputadorLocal\Administrators**. Clique no botão **OK**;
- 3.14. Clique na opção **Read**;
- 3.15. Clique no botão **OK**;
- 3.16. Clique em **OK** novamente.

C – Preparando o ambiente no servidor primário: criação de um database com dados para testes

1. Abra o **SQL Server Management Studio**, conectando-se com a autenticação do Windows;
2. Abra o **Script_05** da pasta **Capítulo_12**. Para tanto, clique em **File**, em seguida, em **Open** e depois em **File** novamente. Abra o **Script_05** com a autenticação do Windows e pressione F5 para executá-lo.

D – Preparando o ambiente nos dois servidores: verificação do serviço SQL Server Agent



Estes passos devem ser executados nos dois servidores.

1. No **Object Explorer**, do lado esquerdo da tela, verifique se o SQL Server Agent está carregado em memória (“startado”). Se esse serviço estiver com uma setinha verde desenhada é porque está ligado;
2. Se o SQL Server Agent estiver desligado, é necessário ligá-lo neste momento. Para tanto, clique com o botão direito do mouse sobre SQL Server Agent e escolha **Start**.

E – Configurando o Log Shipping no servidor primário



Esta etapa deverá ser feita no servidor **Primário**.

1. Do lado esquerdo da tela, no **Object Explorer**, expanda a pasta **Databases**;
2. Clique com o botão direito do mouse sobre o database **Escritorio** e escolha a opção **Tasks**. Em seguida, selecione a opção **Ship Transaction Logs**;
3. Na tela que será exibida, selecione a opção **Enable this as a primary database in a log shipping configuration**;
4. Clique no botão **Backup Settings**;

5. No campo **NetWork path to backup folder** (examples: \\primaryserver\backup), escreva: \\NomeDoServidorPrimario\Logs_A_Enviar;



Substitua a palavra **NomeDoServidorPrimario** pelo nome da máquina em que estamos configurando o servidor como primário.

6. Ainda na mesma tela, no campo **If the backup folder is located on the primary server, type a local path to the folder** (example: C:\Backup), escreva C:\Logs_A_Enviar;
7. Clique no botão **Schedule**;
8. Deixe selecionada a opção **Occurs every** e escreva **2 minute(s)**;
9. Clique no botão **OK**;
10. Clique no botão **Add** para fazer as configurações indicando quem será o servidor **Secundário**;
11. Na tela que será exibida, clique no botão **Connect**;
12. Na tela de conexão, escolha o nome da instância que será o servidor **Primário** e clique no botão **Connect**;
13. No campo **Secondary database**, selecione o banco **Escritorio**;
14. Clique na guia **Copy Files** e, no campo **Destination folder for copied files: (this folder is usually located on the secondary server)**, escreva **C:\Logs_Recebidos**;
15. Clique no botão **Schedule**;
16. Deixe selecionada a opção **Occurs every** e escreva **3 minute(s)**;
17. Clique no botão **OK**;

18. Clique na guia **Restore Transaction Log**;
19. Selecione a opção **Standby mode**;
20. Selecione a opção **Disconnect users in the database when restoring backup**;
21. Clique no botão **Schedule**;
22. Deixe selecionada a opção **Occurs every** e escreva **4 minute(s)**;
23. Clique no botão **OK**;
24. Clique em **OK** de novo;
25. Para encerrar, clique em **OK** novamente e observe o SQL Server realizando as configurações estabelecidas;
26. Assim que tudo terminar com sucesso, clique no botão **Close**.

F – Verificando os resultados

1. Conecte-se no servidor **Secundário** com a autenticação do Windows;
2. Expanda a pasta **Databases** e observe que o database **Escritorio** está com a cor cinza e nele estão aparecendo as palavras **Standby / Read-Only**;
3. Expanda o database **Escritorio** e a pasta **Tables**;
4. Note que a tabela **Pessoa** foi criada juntamente com o database **Escritorio**;
5. Clique com o botão direito do mouse sobre a tabela **Pessoa** e escolha a opção **Open Table**;
6. Note que os dados da tabela original do servidor **Primário** foram copiados para a tabela **Pessoa** no servidor **Secundário**;
7. Feche a query que exibiu os dados da tabela **Pessoa**.

G – Executando comandos DML no database do servidor primário para testes

1. No servidor **Primário**, abra o **Script_06** da pasta **Capitulo_12** e pressione F5 para executá-lo;
2. Note que agora existem 10 registros inseridos na tabela **Pessoa** do servidor **Primário**.

H – Verificando os resultados

1. Aguarde alguns minutos até que o SQL Server faça o backup do transaction log, em seguida, envie-o para o servidor **Secundário** e restaure-o na segunda instância. Este tempo deverá ser de cinco minutos, aproximadamente;
2. Depois de aguardar o tempo solicitado, no servidor **Secundário**, clique com o botão direito do mouse sobre a tabela **Pessoa** e escolha a opção **Open Table**;
3. Observe que, na tabela **Pessoa** do servidor **Secundário**, agora já existem 10 registros, conforme a tabela do servidor **Primário**.

I – Fazendo o Database Standby deixar de ser secundário

1. No servidor **Secundário**, abra e execute o **Script_07** da pasta **Capitulo_12**;
2. Do lado esquerdo da tabela, no **Object Explorer**, clique com o botão direito do mouse sobre a pasta **Databases** e escolha a opção **Refresh**;
3. Observe que o database **Escritorio** não está mais marcado como **Standby** nem como **Read-Only**;
4. Clique com o botão direito do mouse sobre o database **Escritorio** e escolha a opção **Tasks**. Em seguida, escolha a opção **Ship Transaction Logs**;
5. Na tela que será exibida, no campo **Secondary server instances and databases**, selecione o nome do servidor **Secundário** e clique no botão **Remove**;
6. Observe o SQL Server fazer as configurações e clique no botão **OK**.