Administração de Banco de Dados
José Antônio da Cunha
CEFET-RN

ADMINISTRANDO BANCO DE DADOS

Neste artigo procuro esclarecer a importância de duas funções fundamentais da área de TI: A **Administração de Banco de Dados** (DBA) e a **Administração de Dados** (DA).

Muitas vezes confundidas, estas funções precisam ser bem definidas e ressaltadas pela sua importância no contexto dos Bancos de Dados.

Começaremos pelos requisitos básicos que um Administrador de Banco de Dados (DBA) precisa ter para exercer com excelência esta função.

Administrar um banco é, de maneira simplista, instalar, configurar, monitorar e solucionar problemas de um SGBD (Sistema Gerenciador de Banco de Dados). Esmiuçando este conceito, um Administrador de Banco de Dados tem as seguintes responsabilidades:

• Projeto lógico do banco de dados: criação do esquema lógico usando a DDL;

• Definição de checagem de segurança e integridade;

• Decisões de como os dados são representados na base de dados armazenadas;

• Projeto físico da base de dados;

• Definição de procedimentos de recuperação;

• Monitoração do desempenho;

• Contato com usuários para averiguação de disponibilidade dos dados por eles requisitados e ajuda na determinação e resolução de problemas;

• Ajustes apropriados à medida que ocorram mudanças de requisitos.

Para cumprir as responsabilidades supracitadas são exigidos conhecimentos em diversas áreas relacionadas direta e indiretamente com os SGBDs propriamente ditos. Enumeramo-los:

• Arquitetura de computadores: o processo de administração de um SGBD pode exigir o conhecimento da estrutura física de servidores e de como sintonizar hardware e software para obtenção de melhor desempenho e maior segurança;

• Sistemas operacionais: necessidade de conhecer o sistema operacional utilizado pelo SGBD, bem como os conceitos sobre processos, gerência de memória e sistema de arquivos, indispensáveis para a resolução de problemas e definição de procedimentos de recuperação;

• Redes: além do conhecimento básico, é necessário conhecer bem as camadas de rede e aplicação. Conhecer a estrutura da rede nesse nível é de grande importância para monitoração do desempenho;

• Projeto conceitual e lógico de bancos de dados: apesar de não estar envolvido diretamente com o negócio, é necessário conhecer e poder interpretar os modelos de dados que serão criados e armazenados na base de dados, bem como conhecer as implicações que estes modelos podem causar no desempenho de um SGBD.

• Arquiteturas de SGBDs: conhecendo os fundamentos básicos que guiam as implementações dos SGBDs atuais, o administrador tem facilidade no entendimento e questionamento da • arquitetura utilizada pelo SGBD. Muitos conceitos emitidos em treinamentos e manuais específicos de fabricantes, não são completamente entendidos pela falta de uma base teórica do funcionamento de SGBDs.

• Administração de bancos de dados para suporte à decisão ou Data Warehouses : faz-se necessária a reciclagem técnica dos DBAs que passarão a ser responsáveis por grandes repositórios de dados para que os mesmos tenham conhecimento das principais técnicas e soluções disponíveis atualmente para essa nova classe de aplicação. Procuramos esclarecer estas técnicas no nosso curso.

É sempre bom lembrar que administrar um banco é diferente de projetar lógica e conceitualmente um banco. A administração deve prever a utilização do SGBD ao longo de vários anos, garantindo a ausência de problemas físicos futuros que impeçam a disponibilidade dos dados.

Como as bases de dados corporativas estão crescendo intensamente e tornando-se cada vez mais importantes como fontes de informações necessárias à operacionalização das empresas e também como fontes de informações para o processo de tomada de decisão, o Administrador de Banco de Dados deve ser um profissional especialista, capacitado para entender e prestar suporte técnico em cada SGBD utilizado pela organização.

Entendendo como funciona a organização física de um database no SQL Server 2000.

Quando criamos um database, O SQL Server 2000 faz uma pré-alocação de espaço, segmentando o database em páginas de 8kb, numeradas seqüencialmente. Cada conjunto de oito páginas contíguas formam uma unidade lógica maior denominada *extend*. Uma tabela nasce numa extend mista e cresce em extends uniformes, por questão de otimização de espaço.

Quando uma tabela é criada, o SQL Server faz uma consulta nas páginas que controlam extends mistas para obter um endereço de extend com espaço disponível. Da mesma maneira, quando essa tabela precisa se expandir será efetuada uma busca nas páginas que controlam extends uniformes para obter o endereço de uma extend livre (estamos falando de páginas **GAM** e **SGAM** respectivamente).

GAM – Global Allocation Map
SGAM – Shared Global Allocation Map

- Páginas GAM controlam a alocação de extends uniformes;
- Páginas SGAM controlam a alocação de extends mistas.

Essas páginas são criadas no momento da "demarcação" do database, que acontece na sua criação ou no momento da expansão.

Num database, a terceira página será sempre ocupada por uma página GAM e a quarta por uma SGAM, responsáveis por gerenciar as próximas 64.000 extends. A página GAM utiliza um bit para informar se a próxima extend está livre ou não; como existem 8.000 bytes livres numa página, e cada byte controla 8 extends seqüências, chegamos no resultado de 64.000 extends controladas por uma página GAM.

Portanto, o duedo de páginas GAM/SGAM controla até 4GB de dados (64.000 * 64KB) (64 KB é o tamanho de uma extend). Se você criar um database de 5GB, serão encontradas 2 páginas GAM; a primeira será a página de número 3 e a segunda virá após aproximadamente 64.000 * 8 = 512.000 páginas (na verdade, esse número é 511.232, pois são descontados 97 bytes de cada página para controle interno). O mesmo critério vale para as páginas SGAM, ocupando as posições de número 4 e 511.233. Ver **Figura** 1.

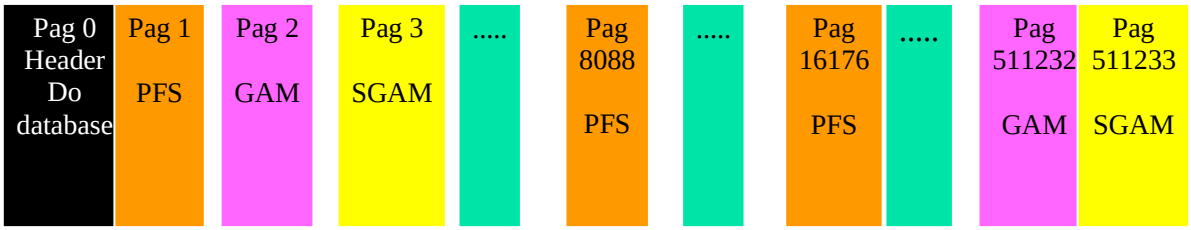


Figura 1: Alocação de páginas de dados no SQL Server 2000.

Além de administrar extends com páginas GAM/SGAM, existe um controle adicional, informando se a página está ou não alocada e seu percentual de utilização. Esse controle é exercido por páginas com o anacrônimo PFS, de *Page Free Space*. Cada página PFS controla 8.088 páginas contíguas num database. A primeira página PFS é a de número 1, logo após a header do database, representada pela página 0. Como mostra a Figura acima.

Existe ainda um controle utilizado para gerenciar as extend utilizadas por heaps e índices, fornecido pelas páginas IAM (Index Allocation Map). Uma página IAM controla 512.000 páginas de uma tabela. Diferentemente das páginas GAM, SGAM e PFS que são demarcadas na criação e/ou alteração de tamanho do database, páginas IAM são alocadas randomicamente (= "on demand") à medida que a tabela (ou índice cresce). As páginas IAM são utilizadas em conjunto com as páginas PFS para orientar o banco nas inclusões.

Assim, quando ocorre um insert numa heap e a página atual já se encontra totalmente preenchida, é efetuada uma busca conjunta nas páginas IAM e PFS para determinar uma página já pertencente a essa tabela para acomodar a inserção. Se não encontrar espaço nas páginas PFS, será efetuada uma requisição na página GAM para uma nova extend.

Observação: tabelas com índices cluster não se orientam com base nas páginas IAM, pois as inserções não são baseadas na teoria de "onde existe espaço", mas sim na chave do índice cluster.

Índices na Otimização de Consultas

Criar *índice* eficiente não é uma tarefa simples; requer conhecimento das consultas (queries) em execução e dos diferentes tipos de índices disponíveis.

Estrutura interna de um índice

Índices são estruturas que possuem algoritmos otimizados para acessar dados. Assim como nas tabelas, páginas de índices também ocupam espaço físico. O corpo de um índice é formado pelas colunas da tabela cujos dados se deseja classificar seguido de uma

referência conhecida como “ponteiro”, que serve para localizar a chave na página de dados da tabela.
Nota: o índice cluster não utiliza ponteiro.

Índices no SQL Server 2000 são construídos sobre estruturas denominadas árvores balanceadas (=“B-Tree”), cujo desenho lembra o esqueleto de uma pirâmide. A idéia desse algoritmo é fornecer um modelo de pesquisa que agilize o processo de busca, efetuando um número reduzido de leituras nas páginas do índice para que se obtenha a localização da chave pesquisada.

Fazendo uma analogia com um livro, quando procuramos por determinada palavra num livro, localizamos a(s) página(s) desejada(s) através de uma busca em seu índice. Se fossemos ensinar alguém como procurar a palavra “ADMIN” num livro de SQL Server, provavelmente ensinaríamos alguma coisa assim:

1. Localize o índice remissivo no final do livro;
2. Procure o bloco de palavras que iniciam pela letra “A”;
3. Efetue uma leitura seqüencial nesse bloco até localizar a palavra desejada.

A **Figura** 2 na página seguinte, ilustra um processo de busca envolvendo a mesma pesquisa anterior numa árvore **B-Tree** de um índice não-cluster. O processo tem início numa página-mestre conhecida como “root page”, procurando pela maior chave da página cujo valor é menor ou igual à palavra pesquisada. Em nosso exemplo, a primeira palavra cujo código alfabético é menor ou igual à “**ADMIN**” é “**ACESSO**”, portanto seguiremos nessa direção até a página de número 2, localizada num nível intermediário conhecido por “non leaf level”. A busca é finalizada no nível folha ou “leaf level page”, onde encontramos a referência para a página de dados onde se localiza a palavra “ADMIN”.

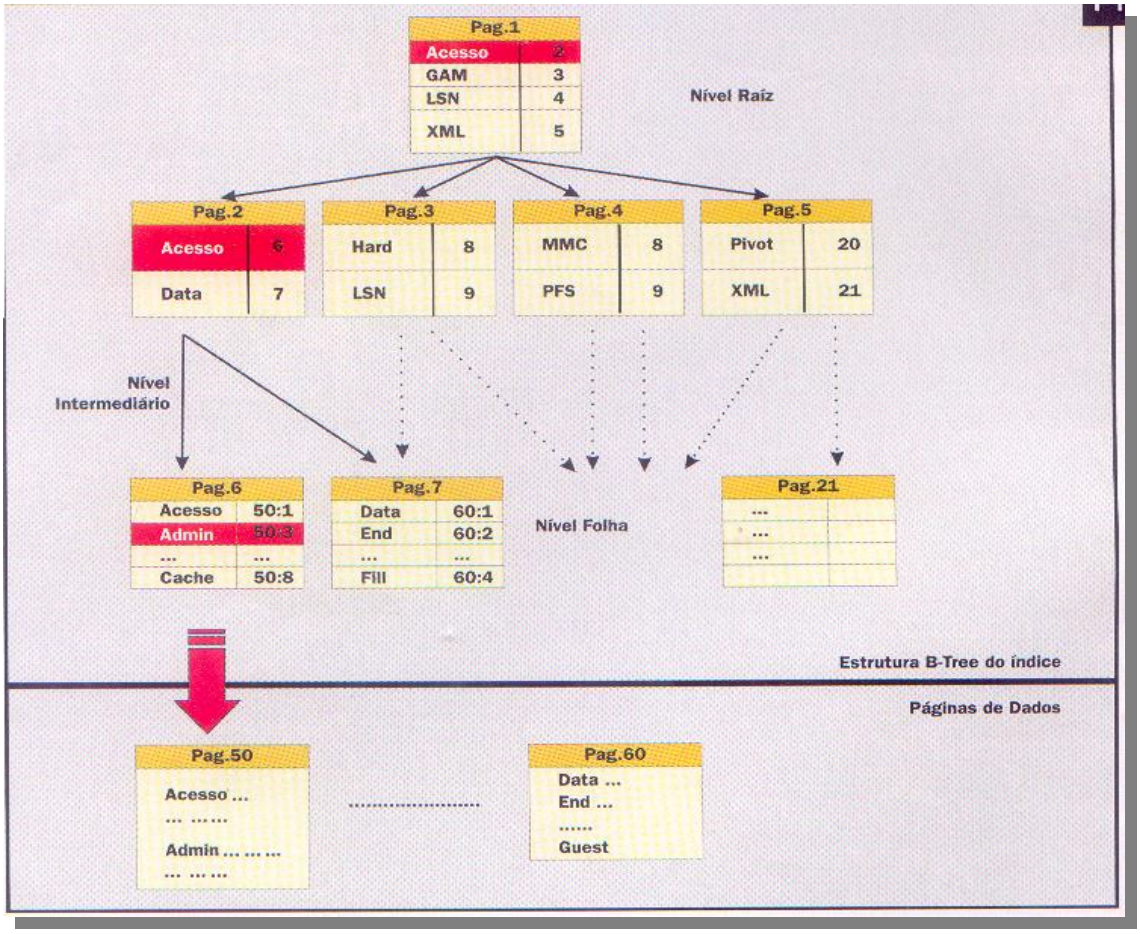


Figura 2: Estrutura de índices no SQL Server 2000.

Tipos de índices existentes no SQL Server 2000

Existem dois tipos básicos de índices:

1. Cluster
2. Não-cluster

Índices cluster impõem uma organização na própria página de dados da tabela, fazendo com que permaneçam classificadas de acordo com a composição de sua chave.

Portanto, se você executar o comando a seguir:

Select * from Northwind.dbo.Orders

Irá notar que os pedidos são ordenados pela coluna OrderID que faz parte do índice cluster PK_Orders. Podemos então afirmar que o leaf level de um índice cluster

representa a própria página de dados da tabela, descartando a utilização de ponteiros para páginas de dados.

Já os índices não-cluster possuem estruturas própria, mantendo-se vinculados às páginas de dados pela utilização de ponteiros.

A tabela SysIndexes é responsável pelo armazenamento dos metadados do índice. Nessa tabela localizamos o nome do índice, uma indicação de seu tipo (cluster ou não-cluster), o número de páginas utilizadas, o número de alterações desde que o último cálculo de estatísticas foi executado etc.

Tabelas sem índice cluster – conhecidas por heaps – possuem uma linha em SysIndexes para IndId=0. Se uma tabela possui índice cluster, este será indicado por IndId=1. Porntanto, se você quiser listar as tabelas que não possuem índice cluster em seu database, basta selecionar as entradas em SysIndexes para IndId=0.

- O termo **cluster index scan** – é utilizado para especificar varreduras seqüenciais nas páginas de dados de uma tabela que possui índice cluster. Nesse caso, a página inicial da tabela encontra-se em SysIndexes para IndId=1.
- O termo **table scan** – é utilizado para especificar varreduras seqüenciais nas páginas de dados heaps. Nesse caso, a página inicial da tabela encontra-se em SystemIndexes.FirstIam para IndId=0.

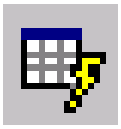
Páginas de dados de tabelas com índice cluster são "ligadas" uma às outras, isto é, no cabeçalho de cada página são encontradas referências à página anterior e posterior (= Next/Previous Page). Para um processo efetuar uma leitura seqüencial numa tabela com índice cluster – conhecida como cluster index scan – precisará apenas localizar a página inicial em SysIndexes.Root; as páginas seguintes estarão encadeadas.

Já em **heaps** o processo é diferente pelo fato das páginas de dados não possuírem ordenação. Pode-se iniciar um lote de inserção numa página localizada "no meio da tabela", utilizando espaço gerado por uma série de deleções e terminar o processo "no fim da tabela", alocando-se uma nova extend.

Como já vimos, em heaps as páginas não são ligadas uma as outras. Então, para varrer as páginas pertencentes a uma heap, o SQL Server utiliza páginas especiais – denominadas páginas IAM – que controlam as páginas utilizadas por uma tabela. Portanto, num processo de leitura de um heap o SQL Server 2000 se norteia pelas páginas **IAM**.

Criação de um índice passo a passo

Para criar um índice na tabela Orders do banco de dados Northwind no Enterprise Manager, expanda o banco de dados selecionando a opção Tables. Clique com o botão direito sobre a tabela Orders, selecione Design Table e na barra de ferramentas clique em manager Indexes/keys para que a tela apresentada na **Figura 3** obtenha o foco principal.



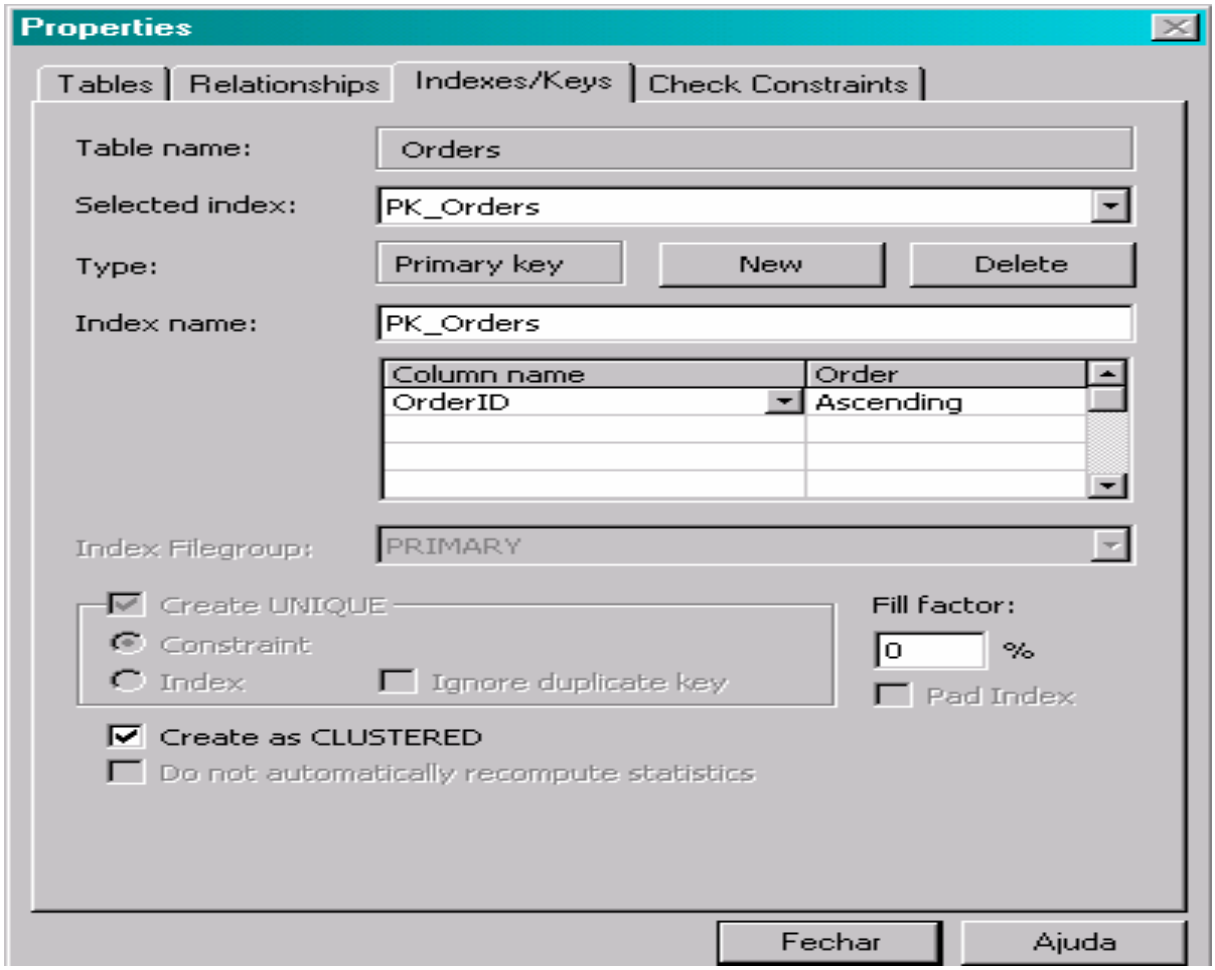


Figura 3: Janela para configuração do índice.

As opções disponíveis na tela de manutenção de índices são:

- **Table Name**: nome da tabela onde se deseja criar o índice.
- **Type**: selecione new para criar um novo índice ou Delete para excluir um índice existente. Os tipos possíveis são Index ou Primary Key.
- **Index Name**: nome do índice.
- **Column Name... Order**: colunas que compõem a chave do índice.
- **Index Filegroup**: indicação do filegroup para criação do índice. Se você não possui discos RAID, uma boa opção para ganho de performance é criar tabelas e índices em filegroups diferentes, localizados em dispositivos distintos.
- **Create Unique**: Unique quer dizer único, que não permite duplicidades.
- **Fill Factor**: indica o percentual de preenchimento das páginas do índice no momento de sua criação. Um fator de preenchimento de 80% informa que será

utilizado somente 80% da capacidade da página para ocupação das linhas do índice. O fill factor atua somente no momento da criação ou reestruturação do índice, não sendo mantido durante os processos posteriores de atualização do índice.

Vale a pena destacar também que:

1. O valor default para fill factor é zero (visível no Query Analyzer sob o comando sp_configure 'fill factor').
2. Fill factor é uma opção avançada de otimização, portanto deve ser utilizada somente naqueles índices onde se observou excessiva fragmentação. Utilizar essa opção de uma maneira genérica para todos os índices do database não é boa prática.

- **Pad Index:**fill factor atua somente no nível leaf level do índice. Assinalando essa opção, o percentual definido em fill factor será propagado para os níveis intermediários da árvore B-Tree.
- **Create as Clustered**: indica que o índice criado será do tipo cluster. Lembre-se que só é possível criar um índice cluster por tabela.
- **Do not automatically recompute statistics**: as estatísticas de distribuição de dados pela chave do índice são essenciais para o otimizador avaliar uma query e, por default, são atualizadas automaticamente após um determinado número de modificações no índice.

Observação: Considerando-se um processo semanal de reestruturação de índices, pode-se dizer que fill factor de determinado índice está adequado à medida que os indicadores do comando DBCC SHOWCONTIG Scan Density e Avg. Page Density (full) se mantêm próximos de 100%. Quanto mais distante de 100%, maior a necessidade de utilização do fillfactor para controle dos custosos page-splits. Portanto, se você encontrar índices de scan density muito inferiores a 80%, experimente estabelecer um pequeno fill factor e reavalie a fragmentação após o mesmo período. Comece, por exemplo, com um índice de 95% para fill factor e vá diminuindo até encontrar seu ponto ótimo.

Exemplo: execute o comando a seguir no query analyser.

DBC SHOWCONTIG (Orders)

Você obterá a seguinte saída, como mostra a **Figura 4:**

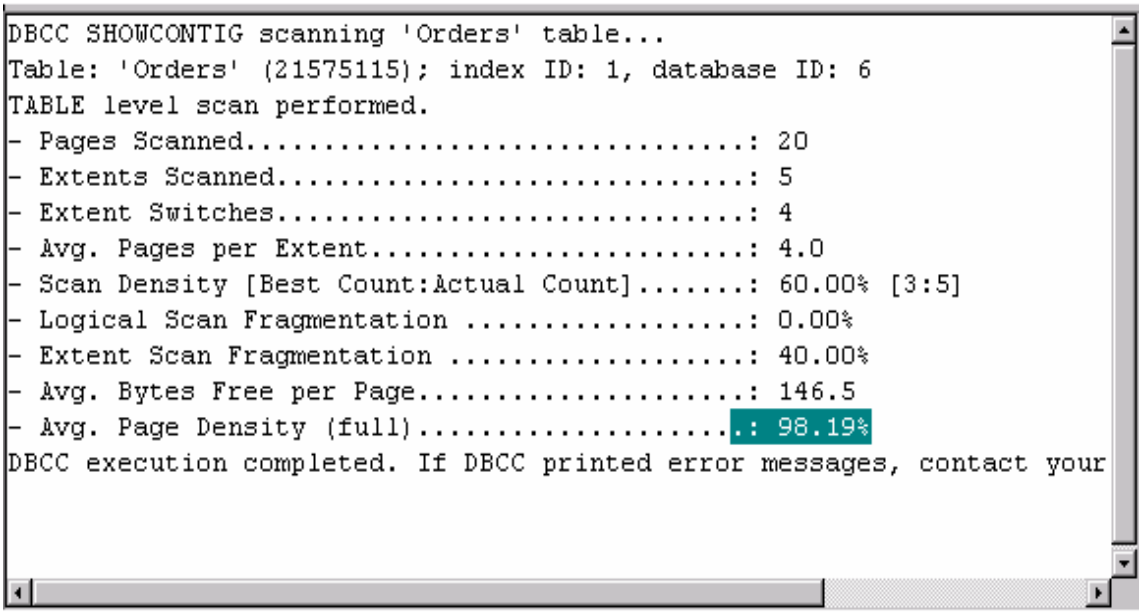


Figura 4: Relatório do comando DBCC SHOWCONTIG na tabela Orders.

Sintaxe do comando DBCC, segundo BOOKS ONLINE:

```
DBCC SHOWCONTIG
[ (
    { 'table_name' | table_id | 'view_name' | view_id }
    [ , 'index_name' | index_id ]
)]
    [ WITH
        {
         [ , [ ALL_INDEXES ] ]
         [ , [ TABLERESULTS ] ]
         [ , [ FAST ] ]
         [ , [ ALL_LEVELS ] ]
         [ NO_INFOMSGS ]
         }
    ]
```

Argumentos:

'*table_name*' | *table_id* | '*view_name*' | *view_id*

É a tabela ou visão para a qual deseja-se checar informações de fragmentação. Se não for especificado, todas as tabelas e visões indexadas (indexed views) no

corrente database são checados. Para obter o ID da tabela ou visão, use a função OBJECT_ID function.

'*index_name*' | *index_id*

É o índice para o qual deseja-se verificar informações de fragmentação. Se não for especificado, o processo utilize o índice base da tabela ou visão especificada. Para obter o ID do índice, use a visão (view) de catálogo **sys.indexes**.

WITH

Especifica opções para tipo de retorno do comando DBCC.

FAST

Especifica que seja feita uma verificação rápida nos índice.

ALL_INDEXES

Efetua a verificação em todos os índices de uma tabela ou view.

TABLERESULTS

Mostra o resultado da verificação em forma de tabela.

ALL_LEVELS

Somente pode ser utilizada em conjunto com a opção TABLERESULTS.

NO_INFOMSGS

Suprime todas as informações de mensagem com nível de severidade de 0 até 10.

Estatística	Descrição
Pages Scanned	Número de páginas na tabela ou índice.
Extents Scanned	Número de extents na tabela ou índice.
Extent Switches	Número de vezes que o comando DBCC move-se de uma extent para outra enquanto o comando atravessa as páginas da tabela ou índice.
Avg. Pages per Extent	Média de páginas por extent
Scan Density [Best Count: Actual Count]	Quanto mais próximo de 100% melhor. Um valor menor do que 100%, significa que

	existe fragmentação.
Logical Scan Fragmentation	Percentage of out-of-order extents in scanning the leaf pages of an index. This number is not relevant to heaps. An out-of-order extent is one for which the extent that contains the current page for an index is not physically the next extent after the extent that contains the previous page for an index.
Avg. Bytes Free per Page	Média de bytes livres na página escaneada. Quanto maior melhor.
Avg. Page density (full)	Average page density, as a percentage. This value takes into account row size. Therefore, the value is a more accurate indication of how full your pages are. The larger the percentage, the better.

A sintaxe T-SQL para a criação de índices no query analyser:

```
CREATE [ UNIQUE ][ CLUSTER | NONCLUSTER ] INDEX index_name
   ON { table | view } ( COLUMN [ asc | desc ] [, ...N ])
[ with < index_option > [ ,...N] ]
< index_option > ::=
     { pad_index |
       FILL FACTOR = fillfactor |
       IGNORE_DUP_KEY |
       DROP_EXISTING | STATISTICS_NORECOMPUTE |
       SORT_IN_TEMPDB
]
```

As opções disponíveis na tela de manutenção de índices são:

- **DROP_EXISTING:** Se droparmos o índice cluster numa tabela que possui também índice não-cluster, todos os índices não-cluster serão reconstruídos, pois o ponteiro desses índices para a página de dados passará a ser RowID. Utilizando a cláusula DROP-EXISTING para que o rebuild nos índices seja efetuado SOMENTE UMA VEZ. (é aplicável somente sobre índices).
- **STATISTIC_NORECOMPUTE**: desabilita a atualização automática das estatísticas do índice, informando ao SQL Server 2000 que as estatísticas do índice serão atualizadas por processo manual. Estatísticas desatualizadas acarretam na escolha de planos de execução ineficientes, portanto sugere-se não utilizar essa opção.

- **SORTE_IN_TEMPDB:** se você possui o TempDB localizado num conjunto de discos separados do filegroup do banco de dados, utilize essa opção para ganho de performance na reconstrução do índice.

Dicas para construir e manter índices eficientes:

- Quanto mais compacto o tamanho da chave do índice, melhor;
- Criar um índice composto ou vários índices?
- Processo de Scan (Clustered Index Scan ou Table Scan) em tabelas com grande número de linhas representam gargalho de execução. Fique atento para isso.
- Procure criar sempre um índice cluster em suas tabelas.
- Bases OLTP são responsáveis por um grande volume de acessos pontuais. Nesses casos, procure criar PK´s clusterizadas e curtas.
- Em bases destinadas a consultas, reserve o índice cluster para colunas que são acessadas por range. (Notas → data)
- Se sua base de dados é utilizada tanto para operações on-line como para consultas diversas, use o bom senso: se for interessante privilegiar os processos on-line, opte por clusterizar as PK´s. se for interessante privilegiar os relatórios, reserve o índice cluster para aquelas colunas que são pesquisadas com cláusulas between, order by etc.
- Não crie índices em colunas com baixa seletividade. Colunas com alto grau de duplicidade não são uma boa escolha para índices não-cluster em função do alto custo.
- Não crie índices em tabelas com pequeno número de linhas.
- Mantenha as estatísticas atualizadas. Mantenha as opções Auto-Create/Update Statistics ligadas.
- Crie rotinas de indexação periódicas. Rotinas de indexação são fundamentais para garantia de performance. Não se esqueça delas.
- Utilize o Profiler como ferramenta de apoio no rastreamento de queries com longo tempo de execução. Aproveite a oportunidade para criar índices mais eficientes ou mesmo dropar índices inúteis.
- Utilize o Index Tunning Wizard como ferramenta de apoio para tuning de índices.
- Ao criar índices compostos, mantenha a coluna mais seletiva no primeiro nível da chave.
- Dê preferência por índices baseados em colunas numéricas em oposição a colunas char ou varchar. Índices baseados em colunas numéricas são mais eficientes.
- Não crie índices em duplicidade. Um erro bastante comum é criar índice com a mesma estrutura de outros já existentes. Habitue-se a executar um **sp_HelpIndex** para confirmação dos índices existentes.

Observações:

- Índices devem ser criados para agilizar a performance do sistema como um todo, mas freqüentemente nos esquecemos disso. Sub-avaliamos o impacto da criação de

índices na performance geral do sistema, e aquilo que foi concebido como objetivo inicial de ganho de performance resulta mais em um ponto de contenção.

- Otimizar um processo pode significar eliminar um índice ineficiente, implementar novos filtros ou alterar os parâmetros da cláusula join das queries em execução. Devemos sim considerar a criação de índices como recurso de otimização, mas numa análise conjunta com todos esses fatores.

Otimização e Tunning

Com o passar do tempo, as tabelas tendem a adquirir fragmentação – os dados que inicialmente ficavam próximos se tornam "espaçados" (caderninho de agenda).

Conceitos sobre armazenamento de dados

No SQL Server 2000, o armazenamento é feito em estruturas físicas conhecidas como "**páginas**". Páginas constituem a unidade básica de **I/O**, possuem tamanho fixo de 8KB e são exclusivas para cada objeto (duas tabelas não podem ocupar a mesma página). Por questão de otimização, páginas são agrupadas em unidades lógicas denominadas "**extents**". Uma extent corresponde a 8 páginas (64KB) e normalmente é utilizada para alocação de espaço para tabelas e índices. Observe que extents são alocadas para um mesmo tipo de página; dessa forma, páginas de dados e de índices são alocadas em extents distintas.

Na verdade uma página não comporta um registro de 8192 bytes (=8KB). Desse montante, devem ser descontados 96 bytes destinados à header da página e 36 bytes para controles de log, resultando em 8060 bytes. Desses 8060 bytes, ainda devem ser descontados 60 bytes para controles internos de colunas de tamanho variável (varchar, nvarchar), chegando então em 8000 bytes.

Tipo de Página	Função
Data	Armazenam dados de tipos diferentes text, ntext e image
Index	Chave dos índices, com ponteiros direcionados para as páginas de dados.
Text and Image	Armazena dados do tipo text, ntext e image.
Page Free Space (PFS)	Controla os espaços livres nas páginas.
Global Allocation Map (GAM)	Controla a alocação de extends.
Shared Global Allocation Map (SGAM)	Controla a alocação de extends mistas pelos objetos.

Index Allocation Map (IAM)	Controla as extends utilizadas por "heap tables" ou índices. Todo objeto no momento de sua criação é registrada numa página IAM e em pelo menos uma extend mista.

Tabela 1 Principais tipos de páginas encontradas num database

Observação: Um objeto nasce, cresce até 8 páginas em extents mista, e passa para extents exclusiva.

- Tabelas constituem a base do modelo relacional para o armazenamento de informações. São formadas por registros que estão fisicamente alocados em páginas que por sua vez estão alocadas (logicamente) em extents. O tamanho de um registro não pode exceder o tamanho de uma página

- Registros podem ser gravados de maneira ordenada ou aleatória. Para que os registros sejam gravados fisicamente de forma ordenada (por exemplo, em ordem de nome na tabela "Cliente"), é necessário, a construção de um índice especial, conhecido por cluster. O índice cluster é a própria tabela, não existindo portanto uma estrutura à parte para guardar informações relativas a ordenação.
- Em virtude dessa característica particular, tabelas podem conter somente um índice cluster. Tabelas sem índice cluster são conhecidas por "heap".

- Por padrão uma página de dados não possui textos ou imagens. Veja a **Tabela 1**, existem páginas especiais para esses tipos de dados. O campo destinado a imagem armazena um ponteiro informando a página inicial onde reside o objeto. Esse mecanismo traz dois benefícios: o primeiro diz respeito à **otimização**, pois a separação torna o processo de leitura mais eficiente. O segundo diz respeito ao **tamanho**, pois uma estrutura à parte permite armazenar imagens até um limite de 2GB (várias páginas podem ser alocadas para um único objeto).
- O SQL Server permite, através da opção "**text in row**", que sejam gravados imagens ou texto na própria página de dados. Se a maior parte de seus campos BLOB é constantemente acessada e possui tamanho inferior a 8KB, é possível ganhar performance habilitando essa opção. A linha de comando a seguir ativa a opção de armazenamento de imagens de até 512 bytes na própria página de dados:

Exec SP_TableOption Cliente, 'text in row', 512

- Páginas de tabelas com índice **cluster** são ligadas umas às outras através de informações contidas no **header da página** (por exemplo, no header da página 1567 estarão identificadas as páginas 1566 e 1568). Em **heaps**, as páginas alocadas são registradas nas estruturas **IAM**, sem ordenação prévia. Para varrer uma tabela com índice cluster, o SQL Server 2000 acessa a **página inicial**, registrada na tabela de

sistema SYSINDEXEXES. Em seguida, as informações contidas no header de cada página direcionam ao restante da leitura. Para heaps, roteiro de leitura é efetuado através das páginas IAM, num leva-etraz que, para leituras seqüenciais, torna-se menos eficiente.

- Causas da fragmentação:
  - **Ocorrência de "page splits"** – termo utilizado para designar uma divisão de página de índice, cluster ou não cluster para acomodar uma inserção pontual.
  - **Deleção de registros** – causando maior espaçamento entre os dados.
- A recuperação de dados fragmentados requer maior esforço de I/O, portanto devemos trabalhar no sentido de minimizar este problema.

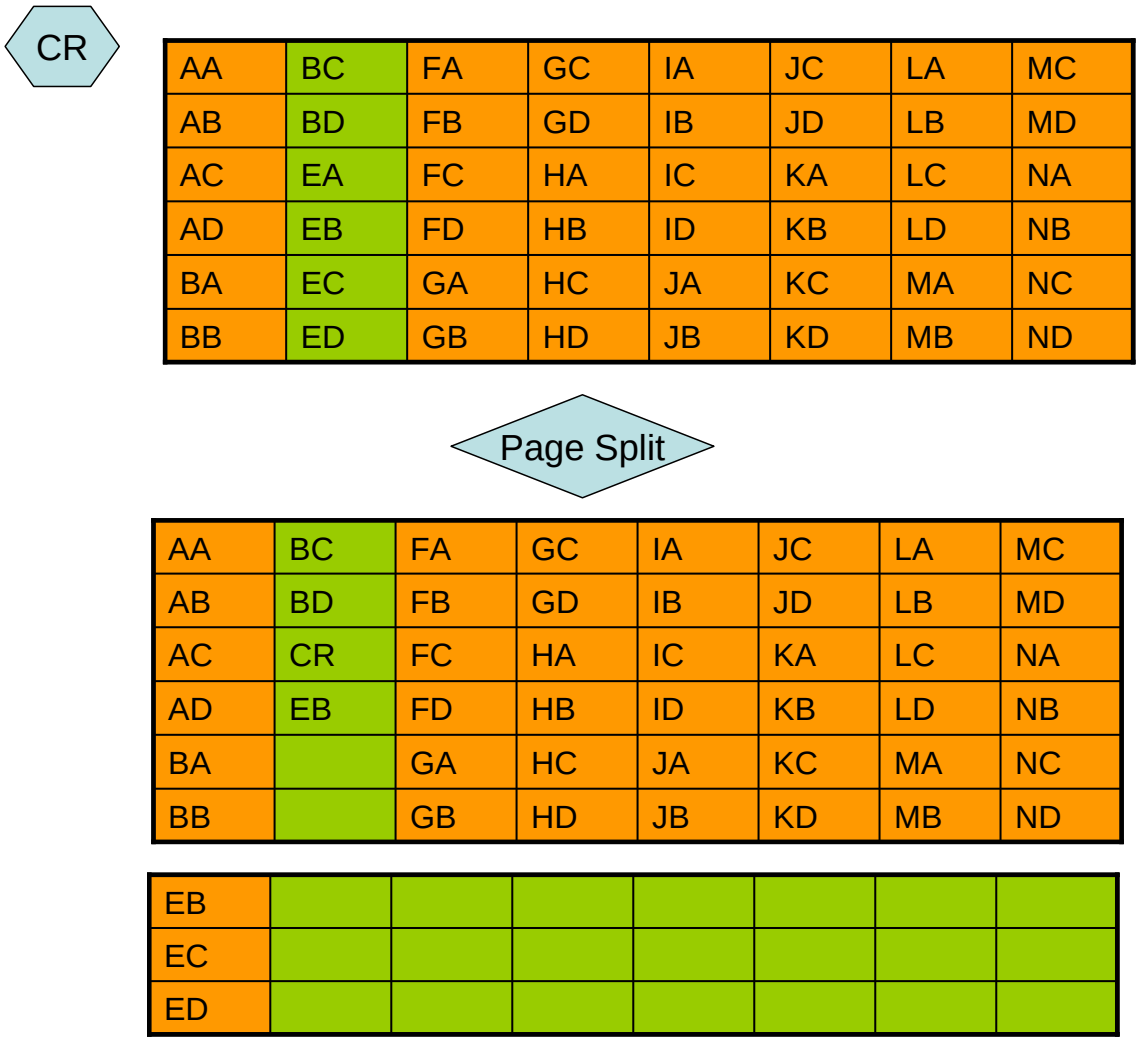


Figura 5: Page Splits gerando uma nova Extent.

O SQL Server 2000 oferece o **comando DBCC ShowContig** para análise da fragmentação em índices. Sua sintaxe é:

DBCC ShowContig (Id da tabela, Id do índice)

Onde

<**Id da tabela**): pode ser obtido pelo comando objeto_id<nome da tabela>
<**Id do índice**>: pode ser consultado através da tabela de sistema sysindexes.

Exemplo:
DBCC SohwContig (Orders, 2)

Execução do comando DBCC ShowContig na tabela "Orders"

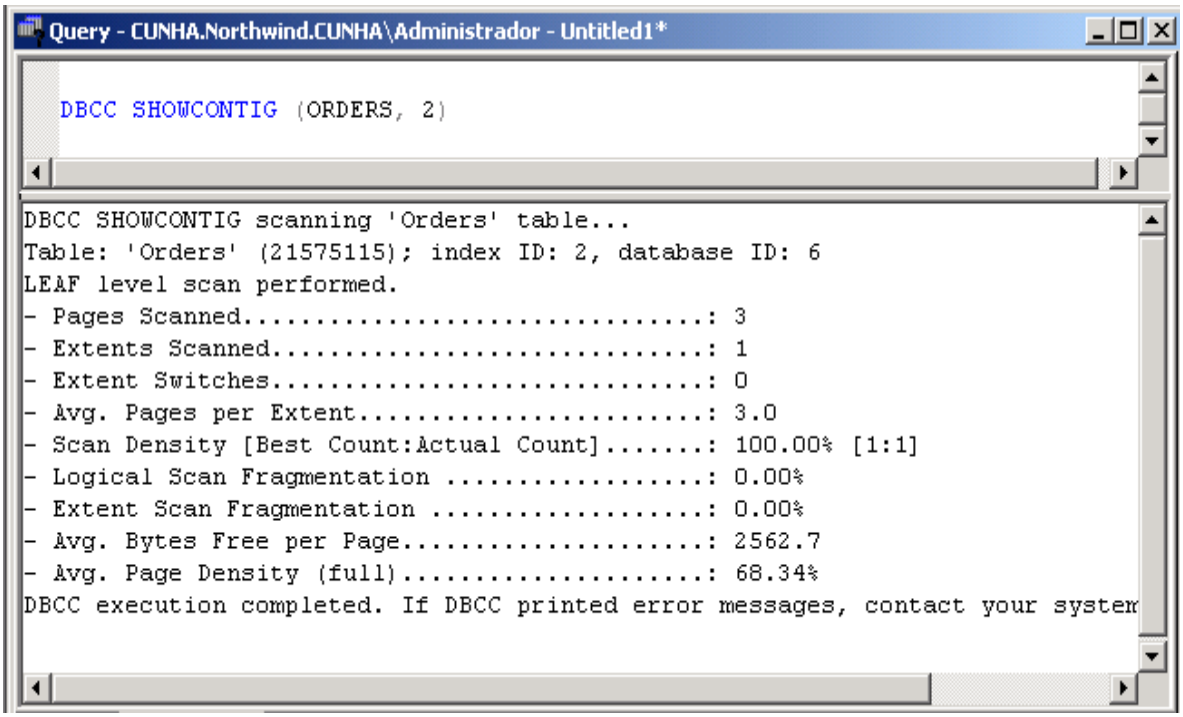


Figura 6: Relatório exibido pelo comando DBCC SHOWCONTIG

O resultado desse comando é interpretado da seguinte forma:

- **Pages Scanned**: número de páginas que compõem o índice analisado.
- **Extends Scanned**: número de extends; é aproximadamente o resultado da divisão de Pages Scanned por 8.

- **Extents Switches**: total de trocas de (páginas que deveriam estar numa mesma extent estão distribuídas em várias extents). Em condições normais deve possuir um valor próximo de Extents Scanned;

- **Avg.Pages per Extent**: número médio de páginas por extent; deve ser próximo de 8.
- **Scanned Density[Best Count:Actual Count**]: densidade das páginas – quanto mais próximo de 100%, melhor. Um valor igual a 75% indica 25% de fragmentação.

- **Logical Scan Fragmentation**: percentual de fragmentação de páginas, utilizado somente para índice cluster.

- **Avg. Bytes Free per Page**: número médio de bytes livres por página; quanto mais próximo de zero melhor;

- **Avg. Page Density(full):** densidade (ou preenchimento) médio das páginas; quanto mais próximo de 100% melhor.

A luta contra dados fragmentados só pode ser combatida com processos de manutenção nos índices, que discutiremos a seguir:

Crie jobs para reindexação periódicas de suas tabelas.

Uma estratégia fundamental para ganho de performance consiste na reestruturação periódica dos índices. Veja a seguir três maneiras de realizar essa tarefa:

- **Drop/Create Index** – O inconveniente é manter o script atualizado para recriar todos os índices de um database;

- **DBCC dbReindex** – Encapsula um DROP/Create para todos os índices da tabela, simplificando a rotina de reindexação. Se alguma falha acontecer, a estrutura anterior será mantida. Possui a desvantagem de estabelecer bloqueios longos;

- **DBCC IndexDefrag** – Elimina a fragmentação INTERNA nas páginas dos índices (não realoca extents). Possui a vantagem de estabelecer bloqueios curtos, sendo possível executá-lo em ambiente de produção.

Listagem 1: Cursor para reindexação de todas as tabelas de um banco de dados.

```
--Batch para reindexar todas as tabelas de um database

set nocount on

DECLARE tabelas CURSOR fast_forward
FOR select name from sysobjects where type = 'u'
DECLARE @nome varchar(80)
OPEN tabelas
FETCH NEXT FROM tabelas INTO @nome
WHILE (@@fetch_status <> -1)
BEGIN
   IF (@@fetch_status <> -2)
     BEGIN
        select '[][][] Reindexando a tabela: ' +@nome
        exec ('dbcc dbreindex ( ''' + @nome + ''')')
     END
     FETCH NEXT FROM tabelas INTO @nome
END
CLOSE tabelas
DEALLOCATE tabelas
```

Algumas considerações sobre a reindexação:

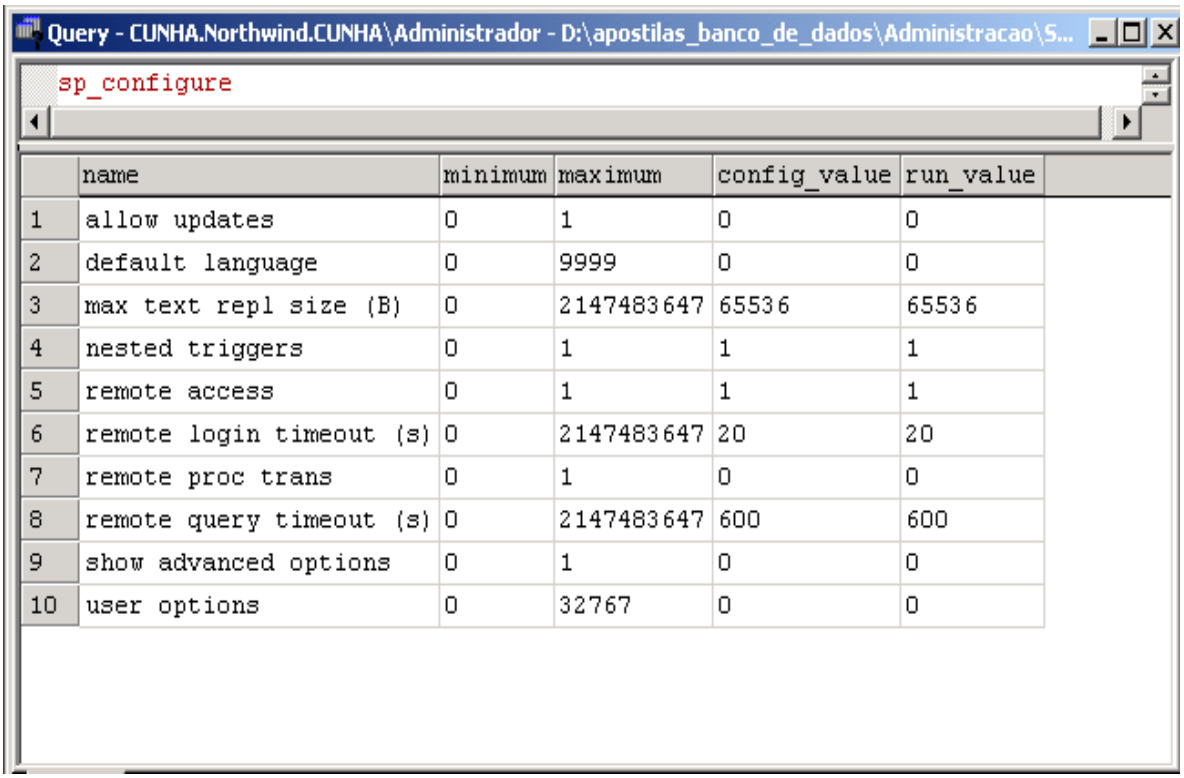
- Se a reindexação de todas as tabelas for custosa (devido ao tamanho por exemplo), você pode optar por reindexar somente as tabelas que possuem fragmentação elevada (Scan Density < 60%), por exemplo).

- Heaps não se beneficiam de processos de reindexação. Reduzir fragmentação em heaps, portanto, significa mover dados para uma área temporária, dropar a tabela, recriá-la e proceder a importação dos dados.

Customizações na configuração padrão

O SQL Server 2000 é bastante otimizado em suas configurações globais. Existem, contudo, alguns parâmetros que podem ser alterados na sua configuração default para efeito de tunning.

O comando **sp_configure**, fornece uma visão detalhada das configurações passíveis de alteração. Veja a Figura 7 a seguir:



	name	minimum	maximum	config_value	run_value
1	allow updates	0	1	0	0
2	default language	0	9999	0	0
3	max text repl size (B)	0	2147483647	65536	65536
4	nested triggers	0	1	1	1
5	remote access	0	1	1	1
6	remote login timeout (s)	0	2147483647	20	20
7	remote proc trans	0	1	0	0
8	remote query timeout (s)	0	2147483647	600	600
9	show advanced options	0	1	0	0
10	user options	0	32767	0	0

Figura 7: Relatório da execução da sp_configure

Para que possamos alterar uma configuração, devemos informar o nome do parâmetro seguido do novo valor. O comando **reconfigure with OverRide** efetiva a alteração, conforme exemplo abaixo:

Exemplo:

Sp_Configure 'show advanced options' , 1

Reconfigure with OverRide

Sp_configure –para confirmar alteração

Algumas opções que podem ser customizadas

- **Max Worker Threads**: pool de threads disponíveis pelo SO para os processos relacionados ao SQL Server. Possui valor padrão de 255, que se adapta bem para grande parte das instalações. Se o número de conexões ativas exceder esse limite, uma thread passará a atender mais de uma conexão (thread pooling). Fique atento para a ocorrência da mensagem "...The working thread limit of 255 has been

reached...". Como sugestão, se o número médio de usuários ativos for superior a 255, altere essa opção e avalie o desempenho do servidor.

- **Priority Boost**: se o servidor não é dedicado ao SQL Server, habilite essa configuração para aumentar a prioridade das threads do SQL Server em relação às outras aplicações.

- **Max Worker Threads**: pool de threads disponíveis pelo SO para os processos relacionados ao SQL Server. Possui valor padrão de 255, que se adapta bem para grande parte das instalações. Se o número de conexões ativas exceder esse limite, uma thread passará a atender mais de uma conexão (thread pooling). Fique atento para a ocorrência da mensagem "...The working thread limit of 255 has been reached...". Como sugestão, se o número médio de usuários ativos for superior a 255, altere essa opção e avalie o desempenho do servidor.

- **Priority Boost**: se o servidor não é dedicado ao SQL Server, habilite essa configuração para aumentar a prioridade das threads do SQL Server em relação às outras aplicações.

Observação: Efetuar tunning em um servidor de banco de dados não é um processo simples, devemos atacar e várias frentes para produzir resultados eficientes. Se, por exemplo, nos concentrarmos em otimização de queries e nos esquecermos de desfragmentar as tabelas, o resultado será modesto.

SQL Server 2000 – Otimização e Tunning

Otimização e tunning de um banco de dados no SQL Server definitivamente não é uma ciência exata: existe o que podemos chamar de "regras de boa conduta" que devem ser implementadas simples que possível; No entanto, o simples cumprimento dessas regras não é garantia de boa performance. O banco de dados utilizado, o comportamento da aplicação, as regras de negócio, a tecnologia de rede e o hardware onde o SQL Server está instalado são alguns dos fatores externos que devem ser levados em consideração e que podem ser decisivos para solução do problema.

Análise do plano de execução de uma query no SQL Server 2000

Como exemplo, analisaremos um comando SELECT executado no database Northwind, (**Listagem 2**). As tabelas Orders e Orders Details tiveram seus índices alterados conforme a **Tabela 2**. o plano de execução gerado (**Figura 8**) , foi obtido diretamente no Query Analyzer. Para isso, selecione o ícone **Query** → Display Estimated Execution Plan, na barra de ferramenta. Os detalhamentos em amarelo na **Figura 9** são obtidos ao posicionar o cursor sobre o respectivo objeto.

```
Select * from [Northwind].[dbo].[Orders] o  INNER JOIN
                [Northwind].[dbo].[Order Details] od
ON o.OrderId = od.OrderID
WHERE o.orderid = 10248
```

Listagem 2: Select executado no database Northwind

Tabela	Nome	Campos	Tipo
Order details	Pk_Order_Details	OrderID, ProductID	Primary key
Orders details	productID	productID	Nom-clustered
Orders	CustomerID	CustomerID	Nom-clustered
Orders	orderDate	OrderDate	Nom-cluster

Tabela 2: Composição dos índices em “Orders” e “Order Details”.

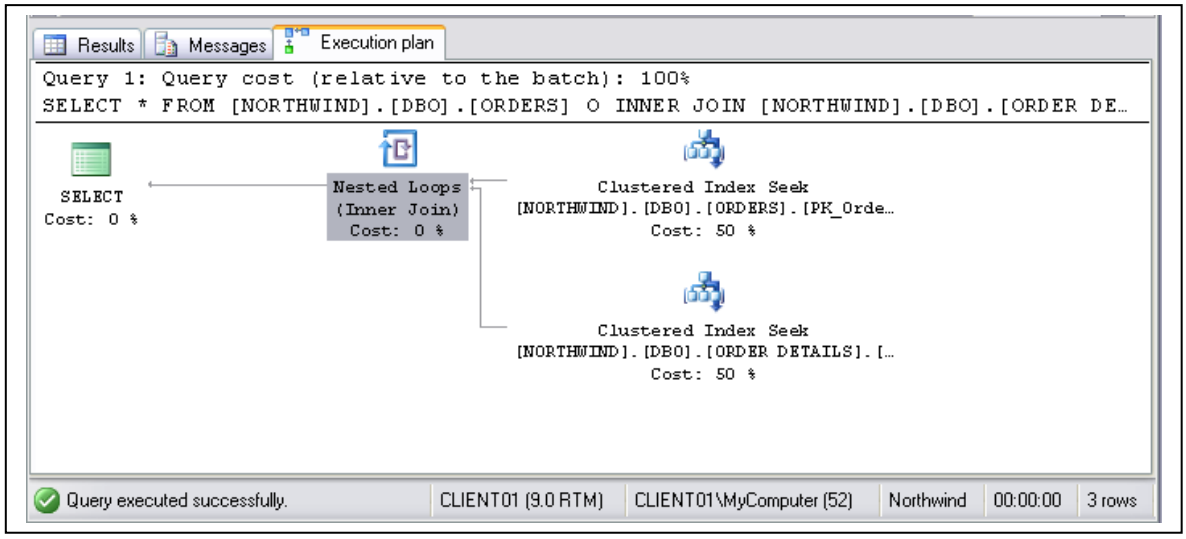


Figura 8: Plano de execução para a consulta da Listagem 1.

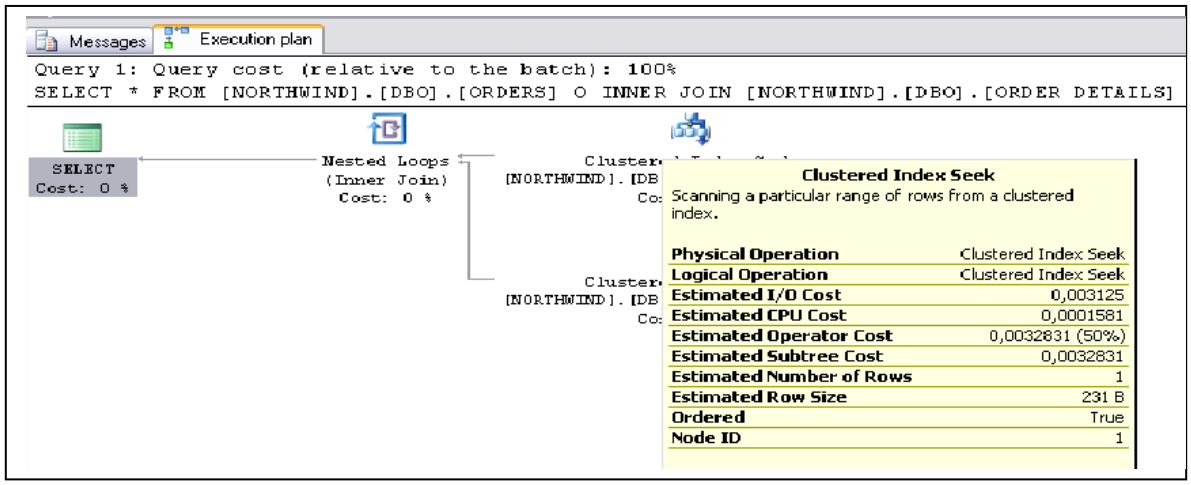


Figura 9: A parte amarela, consegui-se parando o cursor em cima do ícone no plano de execução.

Vamos analisar, através de comentários, o plano de execução do exemplo:

- A leitura do plano de execução deve ser efetuada da direita para a esquerda, de cima para baixo. A espessura das linhas que ligam os objetos é diretamente proporcional ao custo da operação (calculado pela relação número de linhas x tamanho da linha). Portanto, fique atento às linhas mais grossas.
- Cada objeto presente no plano de execução representa uma etapa desenvolvida pelo SQL Server 2000. percorrendo o gráfico, (seek) significa que o otimizador está utilizando índices para pesquisas pontuais. A pesquisa efetuada com (Clustered Index scan) significa que foi utilizado um índice Cluster. Já as pesquisas que utilizam (Index Seek) são realizadas através de índices não cluster. Pesquisas do tipo **SEEK** são bastantes eficientes. Fique atento quando se deparar com o acesso do tipo Table Scan ou Clustered Index Scan, que indicam varreduras seqüenciais por toda a tabela. Acessos do tipo SCAN se devem a pesquisas efetuadas com argumentos insuficientes na cláusula **WHERE**, ou índices não qualificados. Por outro lado, é importante observar que, em alguns casos o uso de Scan é preferível. Por exemplo, para tabelas com pequeno número de registros, a varredura se torna mais econômica do que o esforço adicional causado pela pesquisa no índice.
- Um ponto interessante é que a escolha do índice apropriado para execução do comando SELECT se baseia em estatísticas pré-armazenadas à respeito da distribuição dos dados na tabela. O otimizador irá escolher o método que apresentar o menor custo de I/O para resolver a query, após avaliação das alternativas existentes. As estatísticas de um índice são armazenadas na forma de um histograma, podendo ser visualizada sob o comando:

Dbcc Show_Statistics <nome_da_tabela>, <nome_do_indice>

Um detalhe importante é que essas estatísticas são calculadas para o primeiro segmento (ou primeira coluna) dos índices; portanto, assegure-se de colocar a coluna mais seletiva sempre como o primeiro segmento de um índice. Quanto mais seletiva a coluna menor é o custo de I/O.

- Textos em vermelho nos ícones do plano de execução demonstram estatísticas desatualizadas. Se for esse o caso, proceda à atualização clicando com o botão inverso do mouse sobre o objeto e selecione "**Manage Statistics**".
- O objeto **BookMark Lookup** indica que, para cada registro lido no índice não Cluster, é necessário uma leitura adicional na tabela, pelo fato do índice não contemplar todas as colunas requisitadas pelo comando select. Uma maneira de evitar esse passo adicional é criar índices que contemple todas as colunas requeridas na linha SELECT, recurso conhecido como **covered index**. A criação desse tipo de índice deve ser analisada com critério em bases OLTP, pois os índices degradam a performance em operações de INSERT, UPDATE e DELETE.
- O próximo passo na resolução da query é a escolha do tipo de join. No exemplo o tipo escolhido pelo otimizador foi **Nested Loop**, em função da alta seletividade das tabelas envolvidas.

Nested Loop: o otimizador elege uma tabela, conhecida por **Outer Table**, que servirá de base para leitura seqüencial de registros. A cada registro lido nessa tabela é efetuada uma busca pelo registro correspondente na outra tabela participante do join, conhecida por **Inner Table**. Esse método é bastante eficiente quando uma das tabelas possui quantidade pequena de registros, ou quando o join possui filtros que tornam o resultado pequeno. A tabela com menor número de registros será definida como Outer Table. A Inner Table deverá possuir um índice formado pela(s) coluna(s) que unem as duas tabelas. No exemplo, a tabela Orders representa a Outer Table e a tabela Order Details assume o papel de Inner Table.

Merge Join: se as duas tabelas possuírem índices sobre as colunas mencionadas no join e o número de registros selecionados em ambas for elevado, esse modelo será escolhido. O otimizador recupera uma coluna em cada lista, efetuando a comparação. Em caso de igualdade, retorna as colunas selecionadas. Caso contrário, a coluna de menor valor é descartada e o próximo valor da lista será comparado. O processo se repete até que todas as linhas tenham sido processadas.

Hash Join: se não existirem índices adequados para a igualdade definida no join, esse método será utilizado. Um algoritmo de hash é utilizado para agilizar a combinação. Unir duas tabelas sem índices apropriados ou com baixa seletividade é um fator de queda de performance, portanto investigue esse tipo de ocorrência.

Observação: a escolha do tipo de **JOIN** executado na query é atribuição do otimizador, que foi desenhado para esse fim. Apesar de possível, a utilização de hints para forçar a execução da query por determinado tipo de join é desaconselhável. Se fizermos isso, a query será executada **sempre** da mesma maneira. Uma query executa

por Nested Loop pode ser executada via Merge Join em outro momento, dependendo do volume de dados e dos índices existentes.

Como ilustração, a query a seguir força o plano de execução para Hash Join:

```
SELECT *
  FROM [Northwind].[dbo].[Orders] o INNER JOIN
       [Northwind].[dbo].[Order Details] od
  ON o.OrderID = od.OrderID
OPTION (HASH JOIN)
```

A **Tabela 2** a seguir mostra mais alguns objetos importantes na avaliação do plano de execução.

Icon	Physical operator
	Assert
	Bookmark Lookup
	Clustered Index Delete
	Clustered Index Insert
	Clustered Index Scan
	Clustered Index Seek
	Clustered Index Update

	Collapse
	Compute Scalar
	Concatenation
	Constant Scan
	Deleted Scan
	Filter (clsColumn)
	Hash Match
	Hash Match Root
	Hash Match Team
	Index Delete
	Index Insert
	Index Scan

	Index Seek
	Index Spool
	Index Update
	Inserted Scan
	Log Row Scan
	Merge Join
	Nested Loops
	Parallelism
	Parameter Table Scan
	Remote Delete
	Remote Insert
	Remote Query

	Remote Scan
	Remote Update
	Row Count Spool
	Sequence
	Sort
	Stream Aggregate
	Table Delete
	Table Insert
	Table Scan
	Table Spool
	Table Update
	Top

Tabela 2: Objetos que importantes na avaliação do plano de execução.

 **Assert**: utilizado para verificar condições (integridade referencial, check constraint, entre outras) agindo como uma espécie de filtro para os registros envolvidos na operação.

 **Compute Scalar**: utilizado para retornar saída envolvendo valores calculados (Compute Columns, funções, etc).

 **Index Spool**: indica que foi necessário a criação de tabela temporária no database tempdb para rodar a query. Esse passo muitas vezes pode ser evitado reescrevendo-se o join.

 **Parallellism**: indica que a query está sendo executada em mais de um processador. Em máquinas multi-processadas, o otimizador poderá quebrar queries complexas e executá-las em paralelo, normalmente ganhando performance.

 **Sort**: presente quando a query utiliza order by ou quando o input precisa ser ordenado para resolução do join. No segundo caso, a perda de performance pode ser evitada criando-se um índice adequado.

 **Stream Aggregate**: aparece quando utilizamos cláusulas que agregam valores, como avg, distinct, sum, Max, min ou count.

Dicas para otimização de códigos Transact-SQL

- Limite sua busca restringindo ao máximo o número de colunas na cláusula SELECT. Colunas adicionais, além de consumir mais recursos de I/O e largura de banda, muitas vezes inibem a utilização de índices ou causam buscas desnecessárias na área de dados à partir do índice (**bookmar lookup**).
- Filtre sempre o resultado de suas pesquisas, fornecendo parâmetros de busca que se adequem à estrutura dos índices existentes, obedecendo a ordenação de suas colunas. Por exemplo, para o índice composto Pk_OrderID_Details, formado pelas colunas OrderID e ProductID, é fundamental que uma pesquisa forneça pelo menos o número de OrderID. Fornecer somente ProductID tornará pouco provável o uso do índice pelo otimizador.
- Evite o uso de funções sobre colunas pesquisadas, pois isso inibe a utilização de índices. Exemplo:

Substitua:

WHERE SUBSTRING(ShipName,1,1) = 'M'

Por:

WHERE ShipName LIKE 'M%'

Se não for possível evitar a função, considere a criação de índices sobre colunas calculadas. Veja o exemplo:

```
SELECT DATEPART (month, ShippedDate)
FROM [Northwind].[dbo].[Orders]
WHERE DATEPART (month, ShippedDate) = 7
```

Repare que , se não tiver índice, o otimizador escolherá um Table Scan para resolver a query. Podemos otimizar a consulta criando um índice sobre uma coluna calculada:

```
ALTER TABLE [Northwind].[dbo].[Orders]
ADD Month_OrderDate AS
   DATEPART(month, ShippedDate)


CREATE INDEX IX_Month_OrderDate
  ON [Northwind].[dbo].[Orders](Month_OrderDate)
```

Execute o SELECT:

```
SELECT Month_OrderDate FROM [Northwind].[dbo].[Orders]
  WHERE Month_OrderDate = 7
```

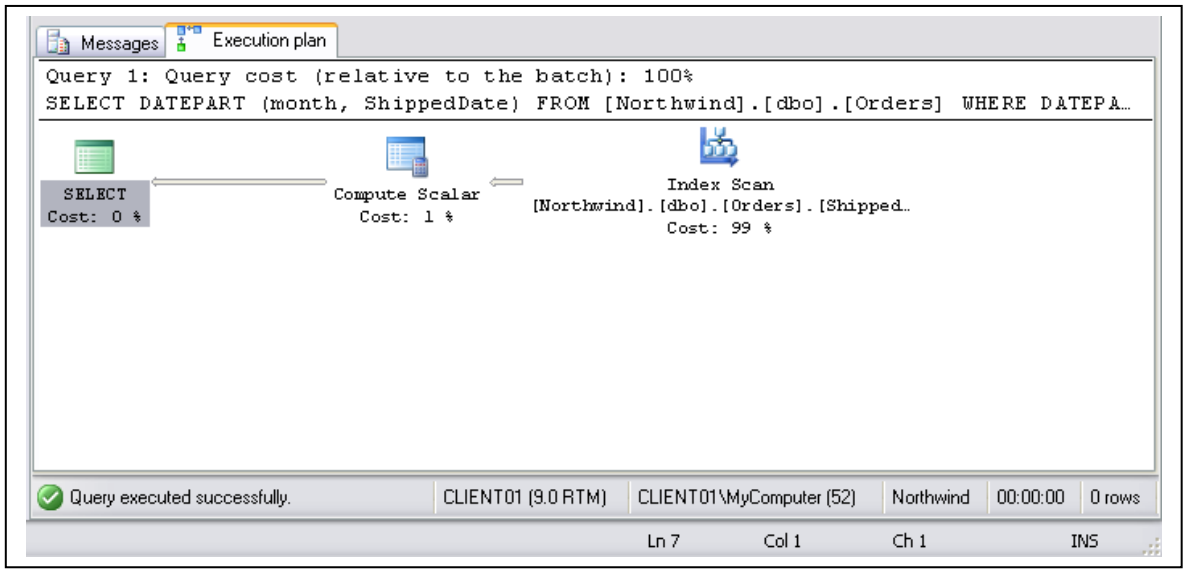


Figura 10: Plano de execução do select anterior.

Observe que o Table Scan foi substituído por um Index Seek! (veja a **Figura 10**).

- Utilize tabela derivadas em oposição à tabelas temporárias. Tabelas derivadas é o resultado da utilização de um comando SELECT dentro de outro SELECT. Um exemplo é o uso de comandos SELECT em cláusulas join.
- Lembre-se que índices existem para comparar igualdades. Evite a utilização de operadores do tipo "<>", "!>", !<" e "NOT". A utilização de lógica negativa inibe o uso de índices pelo otimizador.
- Para fins de performance, considere a utilização de **Indexed Views**. A utilização de views simplifica a programação, mas não otimiza performance, haja visto que seu código é executado de maneira integral a cada solicitação. Indexed Views representa a "materialização das views", geradas através da criação de um índice cluster na própria view.
- Se a query utiliza agrupamentos com filtragem de dados na cláusula HAVING considere a opção de filtragem diretamente na cláusula WHERE, reduzindo significativamente o trabalho do GROUP BY, já que um número menor de registros serão processados.
- Utilize a cláusula like com critério. Lembre-se que o comando WHERE name LIKE 'SQL%' utilizará um índice para a coluna name, se esse índice existir. Já se o índice não existir, o comando realizará um Table Scan na tabela.
- Evite o máximo o uso de cursor no SQL Server. Experimente reescrever o código utilizando outra técnica.
- Sempre que possível, utilize variáveis do tipo **table** em oposição à tabelas temporárias.
- Para monitoramento, utilize o **Profiler** para capturar as queries mais demoradas, analisando seu plano de execução.
- Existem duas configurações de servidor que podem ser utilizadas para limitar o tempo de execução de uma query: query governor cost limit e query wait.

- **Query governor cost limit**: é baseada numa projeção de tempo de execução calculada pelo otimizador. Se o tempo projetado for superior ao limite pré-definido, a query será abortada **antes** de sua execução, gerando o erro 8649.
- **Query Wait**: estabelece um limite de tempo para o SQL Server aguardar pela liberação de recursos de memória para execução da query. Se o limite estabelecido nessa configuração for excedido, a query abortará por timeout.

A alteração desse parâmetro deve ser efetuado com bastante cautela, pois corre-se o risco da query abortada estar inserida em uma transação extensa e ter adquirido muitos locks. Como sugestão, avalie a opção query governor. Implante limites que você considera suficientes para seu ambiente (com boa margem de segurança) e vá reduzindo gradativamente, até chegar ao ponto ideal.

Passos para o otimizador resolver uma consulta (query):

1. Identificação dos argumentos de pesquisa (**SARGS** – Search Arguments) utilizados na cláusula WHERE e colunas mencionadas no join;
2. Seleção do índice apropriado, baseando-se nos SARGS. Os índices são avaliados em função da sua seletividade, utilizando para isso as estatísticas de distribuição. O índice que requer um menor número de leitura para resolver o SELECT será escolhido;
3. Avaliação dos tipos de joins possíveis e ordem apropriada de acesso as tabelas. Isso que dizer que o otimizador definirá a tabela-base do join independente da ordem especificada no comando SELECT. Os comandos a seguir são idênticos:

```
Select o.OrderId from [Northwind].[dbo].[Order Details] od
   INNER JOIN {Northwind].[dbo].[Orders] o
   ON (od.OrdrId = o.OrdrId)

Select o.OrderId from [Northwind].[dbo].[Orders] o
   INNER JOIN {Northwind].[dbo].[Order Details] od
   ON (o.OrdrId = od.OrdrId)
```

4. Seleção do melhor plano de execução, baseado nos custos calculados no item 3.

Análise de bloqueios e deadlocks

Bloqueios são importantes para garantia da consistência de dados em transações. O isolamento fornecido por um bloqueio no SQL Server 2000 permite que uma transação não efetue leituras ou modifique dados que estão sendo utilizados por outra transação.

Existem vários tipos de locks, cada um estabelecendo o isolamento necessário para comandos de manipulação de dados. O tipo de lock (shared, update, exclusive, schema

lock ou bulk update lock) é selecionado automaticamente pelo SQL Server a menos que você utilize um hint, o que não é aconselhável.

O SQL Server trabalha com escalonamento de locks, permitindo que um bloqueio de registro seja promovido para um bloqueio de página ou de tabela. O escalonamento possibilita a economia de recursos (um lock consome 64 bytes de memória), pois os bloqueios de nível menor são liberados.

Como ilustração, imagine uma operação de update envolvendo as 1000 linhas de uma tabela. O que seria mais eficiente: 1000 locks de registro ou um lock de tabela? A segunda opção, já que todas as linhas serão atualizadas.

O problema relacionado a bloqueios advém de seu tempo de duração. Bloqueios curtos são eficientes, bloqueios longos é um transtorno. Entre os fatores que aumentam a duração de bloqueios estão ás transações longas, ausência, excesso ou ineficiência de índices, bases de dados não normalizadas, utilização indiscriminada de cursores, entre outros. Nesses ambientes, as mensagens de erro envolvendo *query timeout* (#1222) ou de *deadlocks* (#1205) tendem a ocorrer com maior freqüência.

Query timeout acontece quando, ao executar um comando de manipulação de dados, aguardamos sua conclusão por tempo superior a um limite previamente estabelecido. Esse limite pode ser definido no objeto de conexão da aplicação front-end ou setado diretamente no batch ou stored procedure através do comando **Set Lock_Timeout**.

Deadlock acontecem quando dois processos ficam aguardando pela liberação dos mesmos recursos. O SQL Server 2000 resolve essa situação finalizando a conexão que consumir menos recursos. Nas **Listagems 3 e 4** seguem exemplos para gerar dois tipos de deadlock: *cíclico* e de *conversão*.

1) Abra duas sessões no Query
2) Na sessão-1, execute o comando abaixo:

```
Begin transaction
   Update [Northwind].[dbo[.[Order Details] set
         Discount=1
   Where orderID=10248 and productid11
```

3) Na sessão-2, execute:

```
Begin transaction
   Update [Northwind].[dbo[.[Orders] set
         employeeid=4
   Where orderID=10248
```

4) Voltando a sessão-1, execute:

```
   Update [Northwind].[dbo[.[Orders] set
         employeeid=5
   Where orderID=10248
   Commit
```

O update acima ficará travado, aguardando a liberação do recurso, bloqueio na sessão-2.

5) Na sessão-2,execute:

```
   Update [Northwind].[dbo[.[Order Details] set
         Discount=0
   Where orderID=10248 and productid11
   Commit
```

Nesse momento acontece o deadlock:
Server: Msg 1205, Level 13, State 50, Line1
Transaction (Process ID 70) was deadlocked on lock resource with another process and has been chosen as the deadlock victim. Return the transaction

Listagem 3: Deadlock cíclico.

1)Abra duas sessões no Query
2)Na sessão-1, execute o comando abaixo:

```
Begin transaction
   Select * from [Northwind].[dbo[.[Order] (holdlock)
    Where orderID=10248
```

3) Na sessão-2, execute:

```
Begin transaction
   Select * from [Northwind].[dbo[.[Orders](holdlock)
     Where orderID=10248
```

4) Voltando a sessão-1, execute:

```
   Update [Northwind].[dbo[.[Orders] set
          employeeid=4
   Where orderID=10248
   Commit
```

O update acima ficará travado, aguardando a liberação do recurso, bloqueio na sessão-2.

5) Na sessão-2,execute:

```
   Update [Northwind].[dbo[.[Orders] set
          employeeid=5
   Where orderID=10248
   Commit
```

Nesse momento acontece o deadlock:
Server: Msg 1205, Level 13, State 50, Line1
Transaction (Process ID 70) was deadlocked on lock resource with another process and has been chosen as the deadlock victim. Return the transaction

Listagem 4: Deadlock deconversão.

Uma das maneiras de monitorar as transações envolvidas em um deadlock á ativar as traces flags 3605 e 1204, que geram informações no log do SQL Server 2000 à respeito do deadlock.

Podemos monitorar também um deadlock com o SQL Profiler, habilitando o evento "Lock: DeadLock Chain", que produz resultados semelhantes às traces flags acima citadas.

Se você não quiser que um processo aguarde indefinidamente pela liberação de um lock mantido em outra sessão, utilize a cláusula "set Lock_Timeout" antes de comandos de manipulação de dados e efetue o tratamento para erros de código #1222.

Minimizar tempos de bloqueios implica no comprimento de algumas regras, a saber:

1) **Mantenha suas transações "enxutas"** – quanto menos código melhor; lembre-se quanto menor o tempo gasto por um bloqueio menor será a possibilidade de ocorrência de deadlocks e travamentos;
2) **Estude a possibilidade de quebrar horizontalmente e/ou verticalmente tabelas com grandes números de registros**. Dados distribuídos permitem maior concorrência, já que os locks estarão dispersos em várias tabelas;
3) **Procure atualizar astabelas nas transações seguindo sempre a mesma ordem,** evitando assim a ocorrência de deadlocks cíclicos;
4) **Evite a utilização de SELECT com hint "holdlocl" seguido de um update.** Essa combinação explosiva é causa freqüente de deadlocks de conversão;
5) **Efetue expurgos periódicos em suas bases OLTP;** não mantenha dados históricos em sua base de dados de produção. Operações em tabelas com grande número de registros tendem a ser mais demoradas;
6) **Não crie índices desnecessários em bases OLTP.** Um índice criado para otimizar uma query representará overhead nas operações de atualização de dados;
7) **Utilize stored procedures em oposição a batchs.** Por estarem residentes no servidor e muitas vezes com planos de execução cacheados, as SP´s apresentam performance superior.
8) **Trabalhe com locks otimistas em situações em que a leitura e modificação de dados representem processos com considerável separação de tempo.**

Observação: Normalmente só pensamos em otimização quando nos deparamos com situações críticas de performance. Otimização é um processo continuo que deve ser observado em todas as etapas de um projeto: inicia-se na modelagem, garantindo estruturas de dados normalizadas e índices bem definidos, continua na escrita dos batchs e stored procedures e prossegue no dia-a-dia do DBA. Através de criteriosas análise do servidor de banco de dados, do rastreamento de queries com longo tempo de duração, da checagem de ocorrências de deadlocks e da implementação de rotinas de reindexação o administrador pode garantir que o banco de dados esteja sempre correspondendo com as necessidades de tempo de resposta da aplicação.

Profiler – SQL Server

Os sistemas normalmente não nascem lentos, mas tendem a ficar mais lentos com o tempo. O aumento do número de usuários, a existência de mais processos concorrentes, o crescimento do volume de informações armazenadas, a falta (ou excesso) de índices e, por fim,a má qualidade do código T_SQL são atores que ocasionam o aparecimento de gargalos e, conseqüentemente queda de performance.

Antes de pensar que o problema "vem de fora" e cogitar em aumentar o poder de fogo do processador, discos ou memória, cabe uma análise mais detalhada dos processos

ativos no servidor de banco de dados. Muitas vezes todo o problema pode ser resolvido com a adição de um índice ou filtro num comando update.

Mas como saber exatamente onde está o problema?

O SQL Server possui um utilitário chamado ***Profile***r, indicado para rastreamento dos eventos processados numa base SQL Server 2000. o Profiler é uma ferramenta de diagnóstico, ou seja, ela nos fornece material para análise. Vale destacar que ela não se propõe por si só a efetuar correções ou qualquer espécie de tuning.

Criando uma trace passo-a-passo

O Profiler é uma ferramenta para criar traces. Uma trace é como uma fotografia dos comandos executados pelo SQL Server 2000 em um determinado intervalo de tempo. Para criar uma trace, selecione Profiler no sub-menu do SQL Server (ver **Figura 1**). Na tela principal do Profiler, selecione File | New | Trace (ver **Figura 2**).

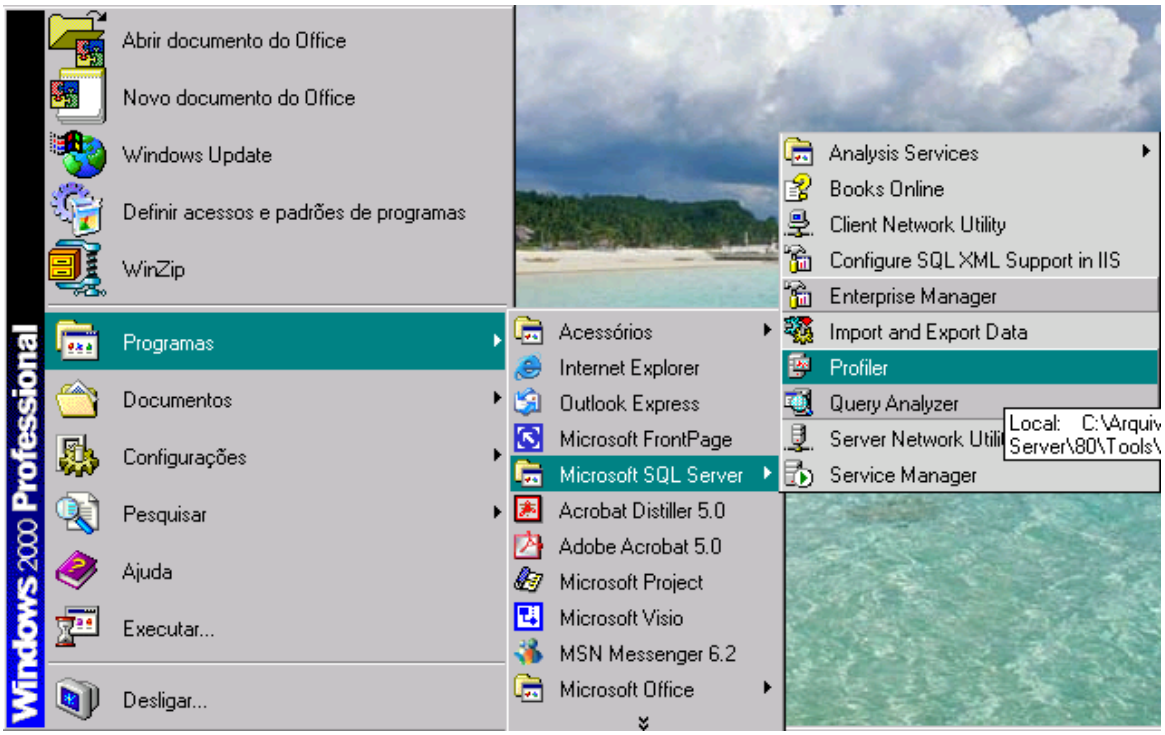


Figura 11 Selecionando o Profiler no Sub-menu do Microsoft SQL Server

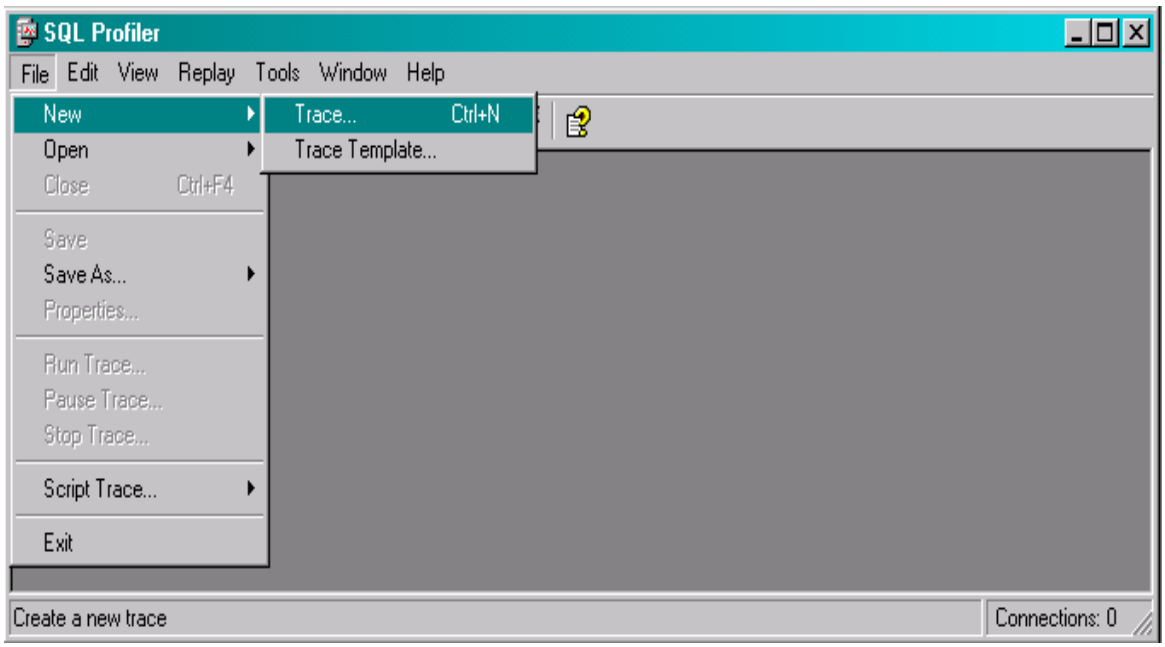
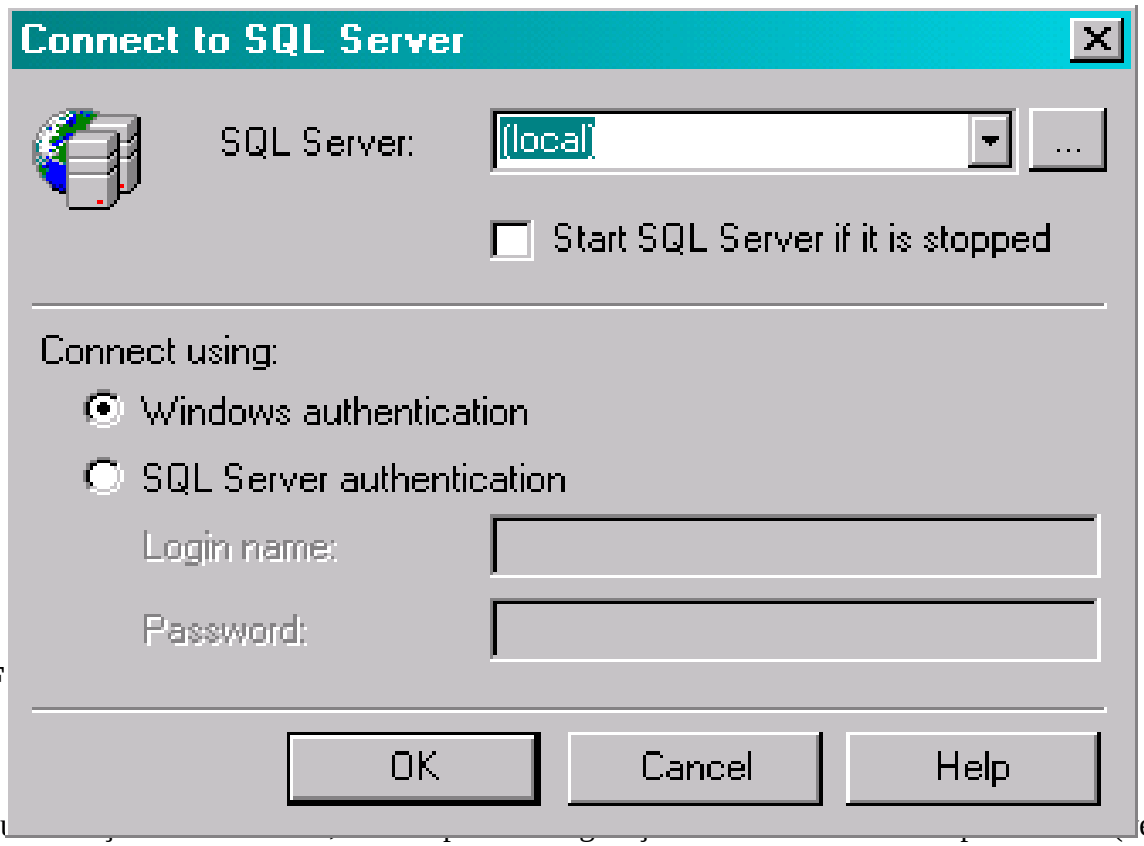
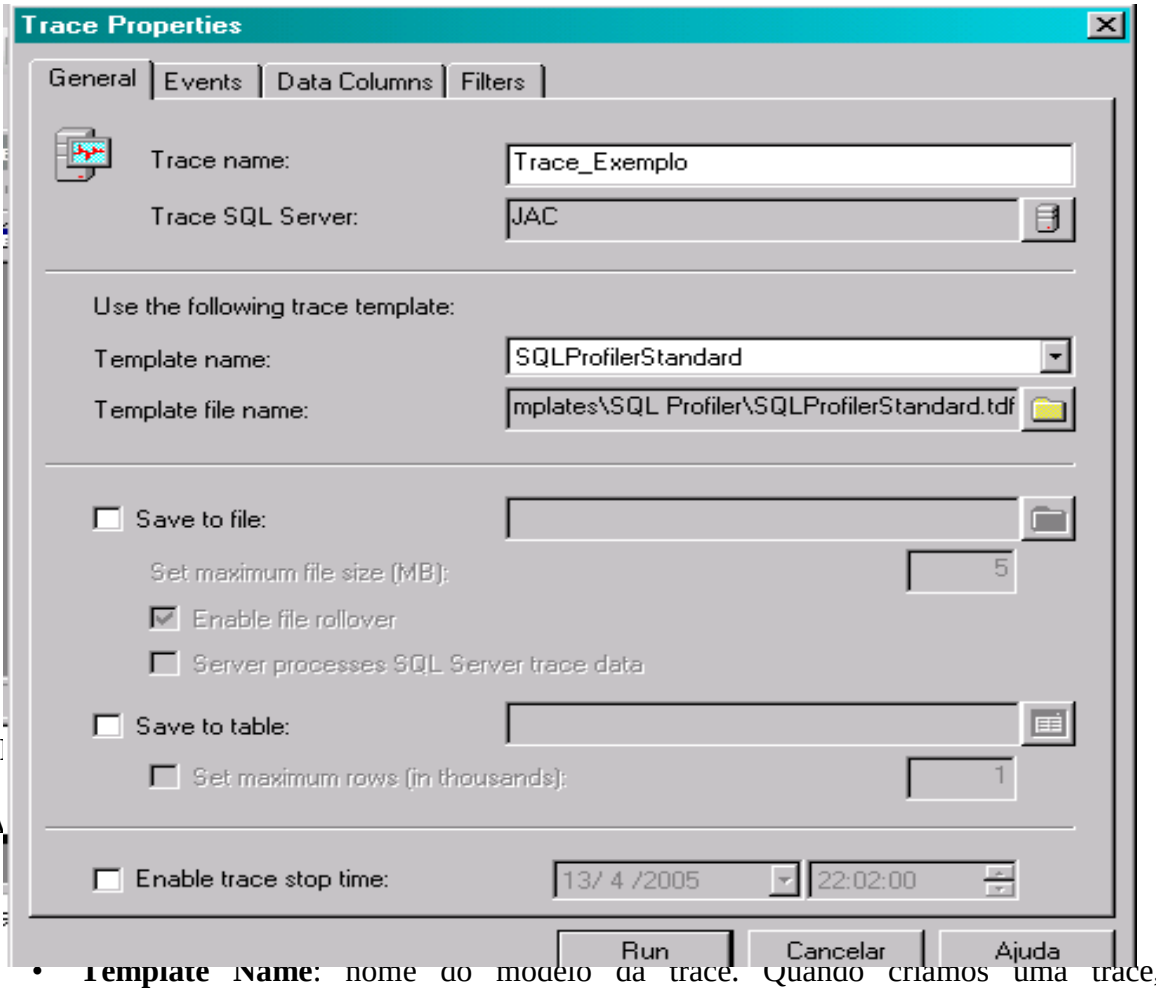


Figura 12 Criando um trace no Profiler

O próximo passo será fornecer uma conta com privilégios de system administrator (SysAdmin) para realizar a trace (veja a Figura 3).



F a
au er
Figura 4).



A

- **Template Name**: nome do modelo da trace. Quando criamos uma trace, selecionamos determinados tipos de eventos que desejamos analisar. Para que não precisemos informar sempre os mesmos eventos ao criar uma nova trace, salvamos modelos chamados templates. Existem alguns templates pré-definidos, o SQLProfilerStandardé um deles. A **Tabela 1** fornece uma descrição resumida dos templates existentes.
- **Template file Name**: caminho do arquivo de template utilizado, cuja extensão é .TDF;
- **Save to file**: grava o resultado da trace num arquivo em disco com extensão .TRC;
- **Set maximum file size**: informa o tamanho máximo do arquivo em disco gerado pela trace. Ao atingir esse limite a gravação em arquivo é suspensa mas o monitoramento continua ativo na tela do Profiler.
- **Enable file rollover**: se o rollover estiver habilitado e o arquivo atingir o limite definido em Set maximum file size(MB), o arquivo em disco será reiniciazado. Neste caso, perde-se o que foi registrado em arquivo até esse momento.
- **Server process SQL Server trace data**: se você algum dia se deparar com a linha de texto em sua trace "... Some events may have been lost...", isto quer dizer que o servidor está muito ocupado e optou por não enviar alguns comandos para sua trace para ganhar algum fôlego de processamento. Habilitando essa opção, você estará forçando o servidor a enviar todos os comandos processados para a trace, mesmo causando perda de performance. É recomendado não utilizar.

- **Save to table**: grava o resultado da trace numa tabela. É mais fácil de depurar, pós podemos colocar filtros ou ordenar da maneira que acharmos mais interessante.
- **Set maximum rows (in thousands):** limira o número de linhas na tabela originada pela trace.
- **Enable trace stop time**: estabelece prazo limite para término da trace.

T1

Nome do Template	Para que serve
SQLProfileStandard	Trace genérica; rastreia comandos executados com sucesso no servidor.
SQLProfileTSQL	Utilize para visualizar os comandos T-SQL startados no servidor.
SQLProfileTSQL_Duration	Utilize para obter uma trace de comandos processados no servidor ordenados por tempo de execução.
SQLProfileTSQL_Group	Lista os comandos startados com sucesso no servidor, ordenados por ApplicationName, NTUserName e ClientProcessId
SQLProfileTSQL_Replay	Utilizada para gravação de traces para posterior replay.
SQLProfile_SPs	Utilizado para visualização dos comandos T-SQL executados internamente nas sp´s.
SQLProfileTuning	Utilizado na geração de eventos para posterior análise pelo Index Tuning Wizard.

Guia events

A guia Events (ver Figura 5), apresenta uma relação de todas as classes de eventos que podem ser monitorados num servidor de banco de dados SQL Server 2000. Nesse contexto, classes são agrupamentos de eventos que possuem uma característica em comum: Temos uma para controlar execução de procedures, outra para gerenciamento de locks, etc. o template SQLProfilerStandard, por exemplo, seleciona automaticamente alguns eventos vistos na **Tabela 2**.

T2

Classe	Evento	Para que serve
Security Audit	Audit Logon	Auditar abertura de sessões no banco.
	Audit Logoff	Auditar encerramento de sessões no banco.
Sessions	Existent Connections	Lista todas as conexões ativas no banco no momento em que a trace é iniciada.
Stored Procedures	RPC:Completed	Lista a execução de sp´s originadas por conexões remotas (ADO, ODBC, OLEDB etc).
TSQL	SQL:Batch Completed	Lista as queries executadas fora do contexto de uma stored procedure.

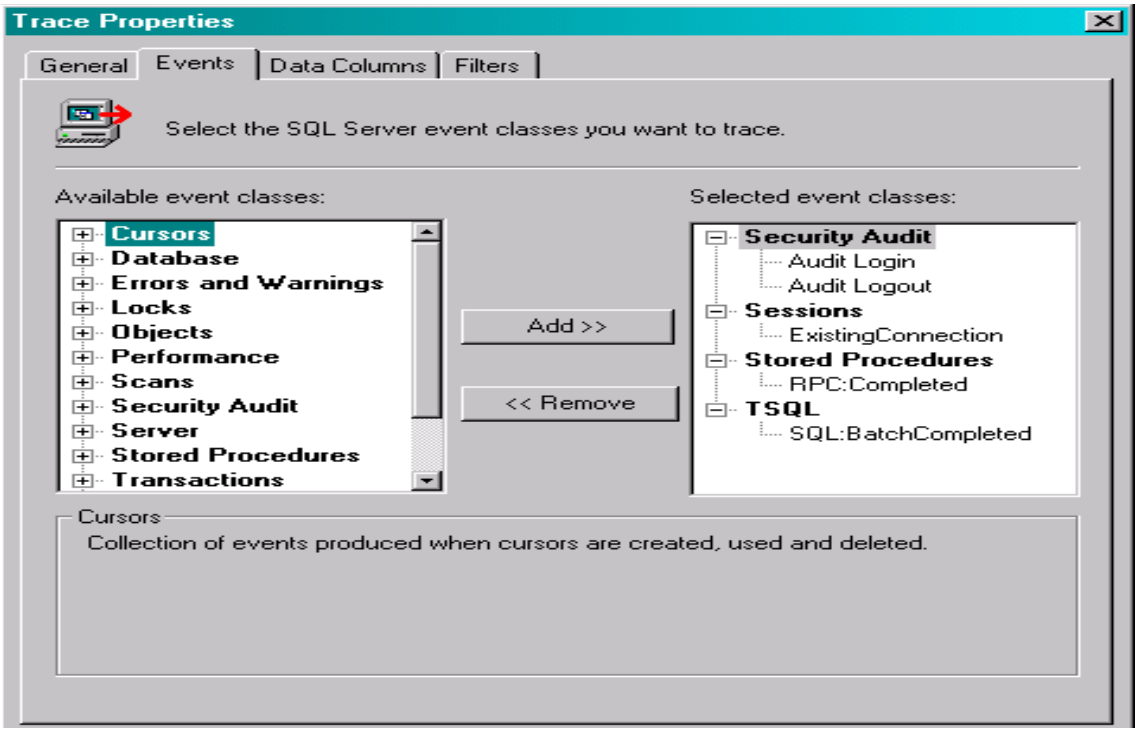


Figura 15 Guia Events

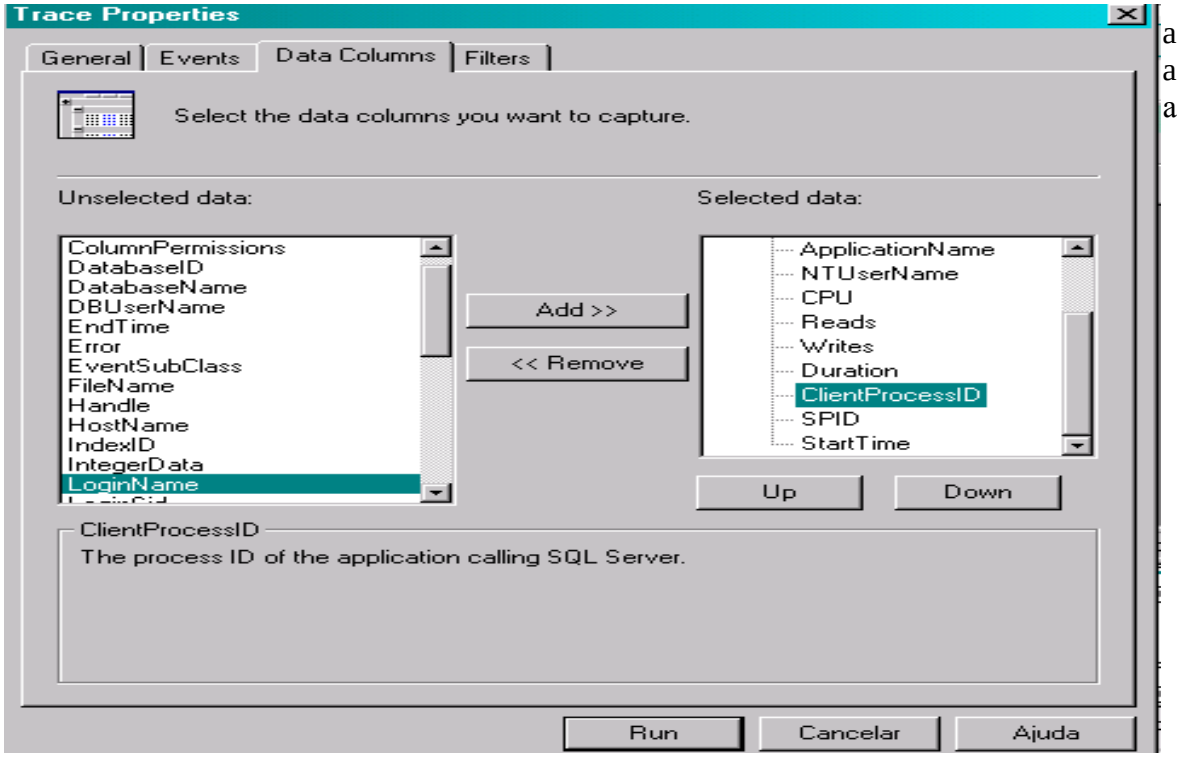


Figura 16 Selecionando colunas que serão visualizadas no Profiler.

Guia Filters

Finalmente a definição da trace na guia Filters (veja Figura 7), utilizada para refino da trace. A aplicação Profiler para mostrar na tela os comandos processados pelo servidor SQL Server, é responsável por uma série de comandos que são também processados pelo engine do banco. Para evitar que esses comandos apareçam na trace, ligamos o filtro ApplicationName Not Like “Profiler”.

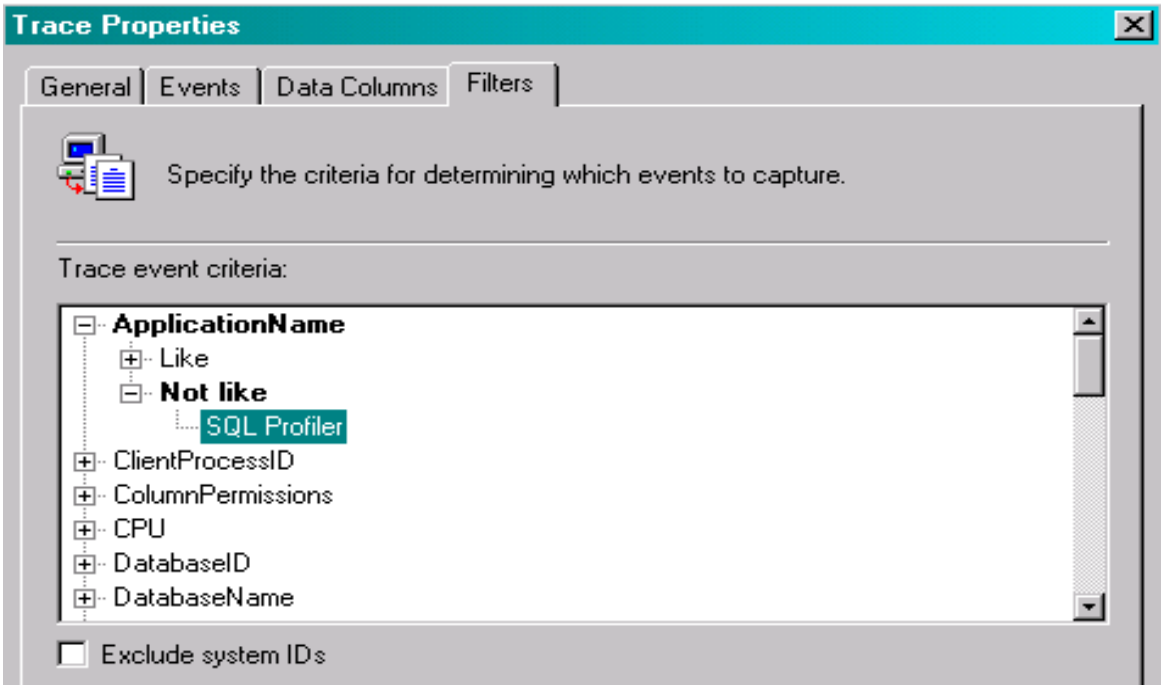


Figura 17 Criando filtros para trace.

Filtros são utilizados para limitar os eventos rastreados na trace, reduzindo o número de linhas afetadas, facilitando nossa compreensão e melhorando o foco de nossa análise. Poderíamos, por exemplo, filtrar os comandos filtrados por um determinado spid. Se desejássemos analisar a execução de uma stored procedure em particular, poderíamos concentrar nossa análise somente na execução dessa sp, utilizando também os recursos de filtros (nesse caso, o filtro ObjectId armazenaria o Id da trace que queremos analisar).

Concluído o processo de definição, clique em RUN para iniciar a trace (ver **Figura 8**).

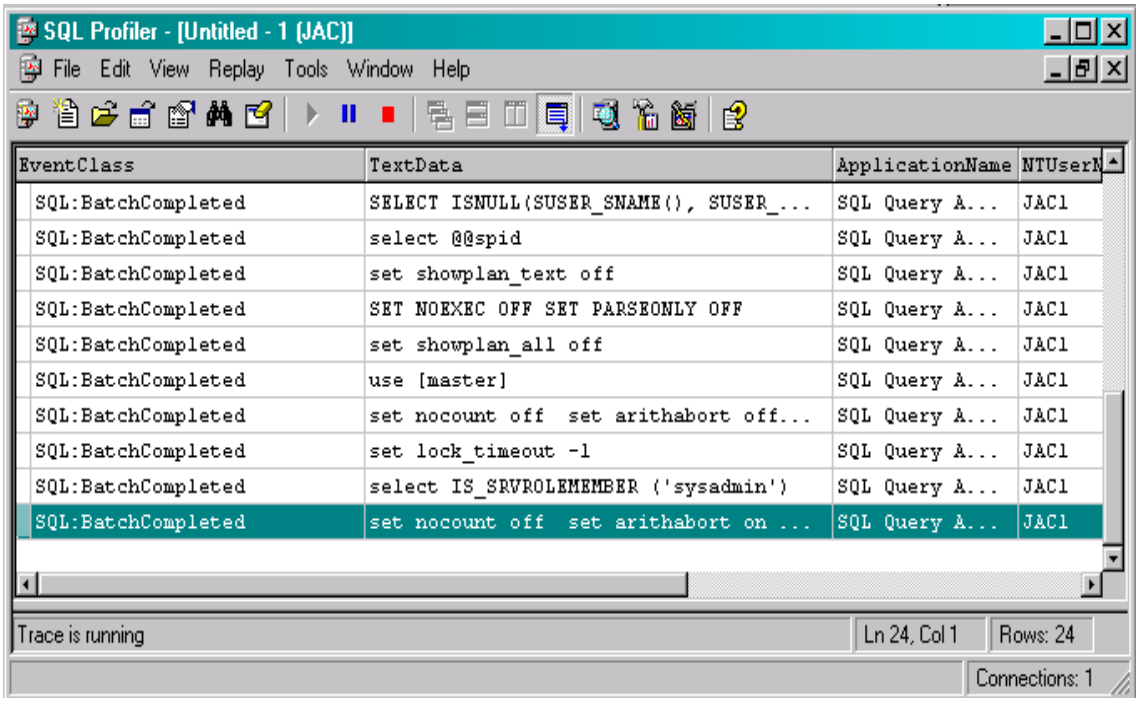


Figura 18: Tela para monitoramento do Profiler

Um resumo do significado das colunas apresentadas na Figura 8 é apresentado a seguir:

- **EventClass**: os eventos rastreados pelo Profiler são agrupados em classes.
- **TextData**: utilizada para visualização do dado coletado na trace. Essa coluna depende do tipo de evento capturado (TCP/IP, evento de conexão).
- **ApplicationName**: nome da aplicação;
- **LoginName**: login do usuário responsável pela execução do comando;
- **CPU**: tempo consumido de CPU para execução do comando (milissegundos);
- **Reads**: número de páginas lidas em memória para executar o comando;
- **Writes**: número de páginas gravadas pelo comando;
- **Duration**: duração do comando(em milissegundos);
- **SPID**: identificação da sessão no SQL Server;
- **Start Time**: horário do início da execução do comando.

Através desta interface é possível:

- Parar a trace: Para isto clique no botão 
- Iniciar a trace: Para isto clique em 

- Iniciar uma nova trace, efetuando toda a parametrização novamente: Para isto clique em 
- Trocar o Template SQLProfilerStandard: Para isto clique em 
- Carregar uma trace previamente gravada em arquivo .TRC: Para isto clique no ícone 
- Carregar uma trace gravada em uma tabela no banco: Para isto clique em 
- Acessar a tela de configurações gerais da trace: Para isto utilize o ícone de propriedades 
- Procurar por uma determinada string na trace que você acabou de gerar: Para isto utilize o binóculo 
- Efetuar uma limpeza na tela: Para isto utilize a borracha 

Agora um teste prático. Com o Profiler ativo, abra uma sessão no Query Analyzer e execute a seqüência de comandos a seguir:

```
Use Northwind
Go
Create procedure stp_Mostrar_Pedido (@OrderId int0
As
   select O.OrderId, O.CustomerId, O.EmployeeId, d.ProductId, d.UnitPrice, d.Quantity
  from Orders O inner join [Order Details] d
         on O.OrderId = d.OrderId
         Where O.OrderId = @OrderId
Return
go
```

Agora um teste prático. Com o Profiler ativo, abra uma sessão no Query Analyzer e execute a seqüência de comandos a seguir:

```
Exec stp_Mostrar_Pedido 10249
go
```

Dirija-se ao profiler e confirme o resultado (veja a **Figura 9** no slide seguinte).

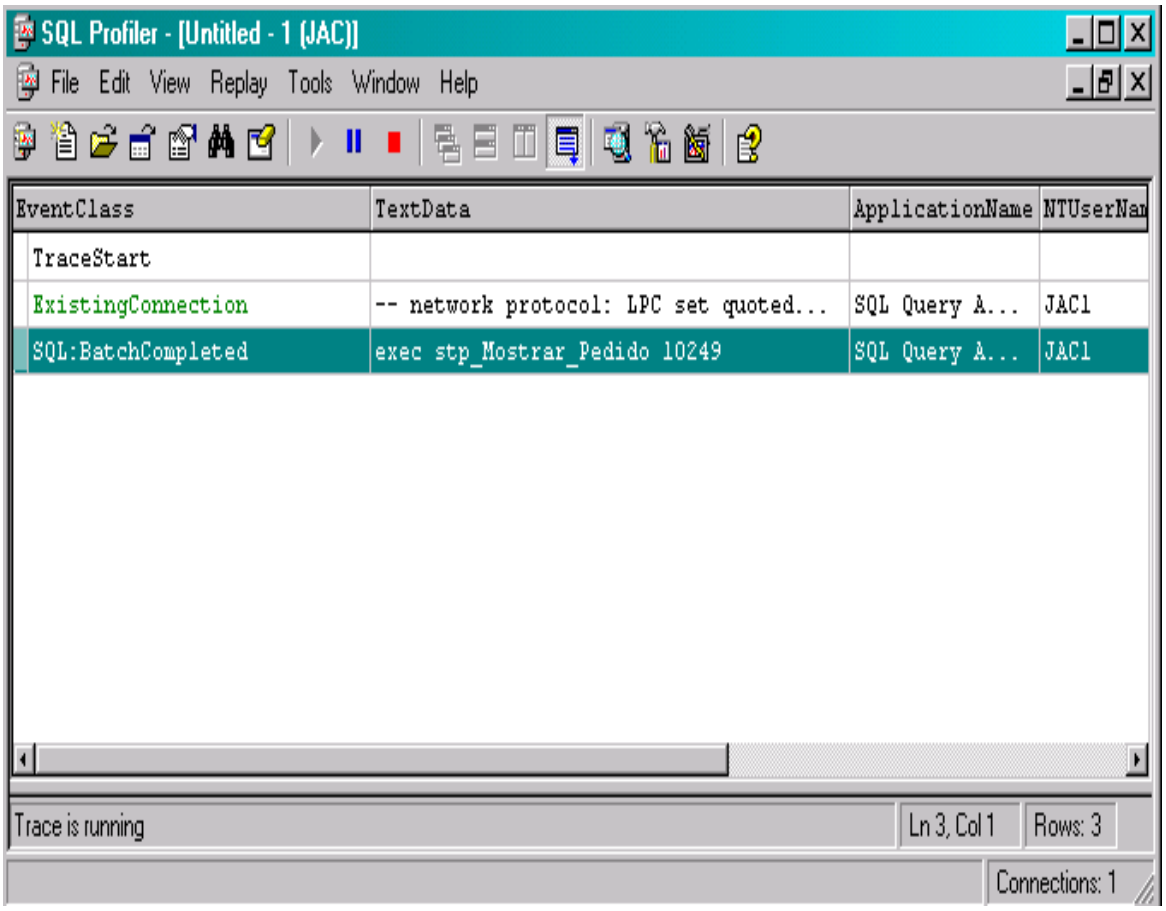


Figura 19: Resultado da execução de comandos na console do profiler.

Gravando a Trace

Para gravar a trace, siga até a opção File da barra de menu e escolha Save AS (ver **Figura 10**).

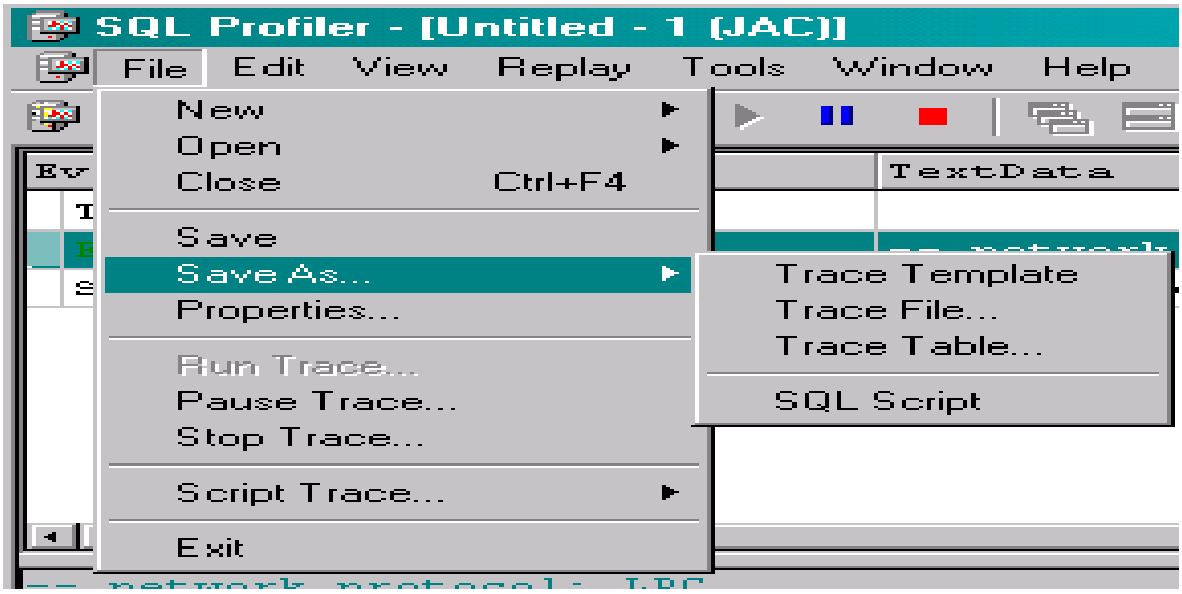


Figura 20: Salvando a trace

As opções disponíveis para salvamento da trace são:

- **Trace Template**: utilize para gerar um template (arquivo com extensão .tdf, de Template Data File);
- **Trace File**: salva um arquivo em disco com extensão .trc com o resultado da trace;
- **Trace Table**: armazena o resultado da trace em uma tabela.
- **SQL Script**: gera um arquivo texto (extensão .sql) com o lote de comandos T-SQL necessários para criar e executar a trace.

Conclusão

Quando o assunto é tuning, o Profiler é uma ferramenta indispensável. Na próxima aula continuaremos esse assunto e nos aprofundaremos nos principais eventos que devem ser analisados, tendo em vista a otimização de processos. Até lá!

Profiler (continuação)

Criando uma trace para análise de performance de um servidor SQL Server

1. Análise preliminar: identificando stored procedures e batchs com baixa performance.

Demora excessiva para conclusão de consultas, deadlocks em demasia e problemas com timeout nas aplicações – esse é o quadro de um servidor com sérios problemas de performance. Nosso objetivo inicial será efetuar um levantamento para determinar quais são os processos mais demorados, cronometrando a execução de batchs e stored ptrocedures nesse servidor.

Passos para realizar esta tarefa:

1) inicie o Profiler.
2) Escolha New Trace na opção File da barra de ferramentas.
3) Na tela de propriedade da Trace, confirme a seleção do template SQLProfilerStandard na guia General. (Esse template captura a execução de batchs(TSQL\SQL:batch Completed e Stored Procedures\RPC:Completed).
4) Os eventos Security Audit e Sessions não são necessários para essa análise, podendo ser removidos. Veja a **Figura 21**.

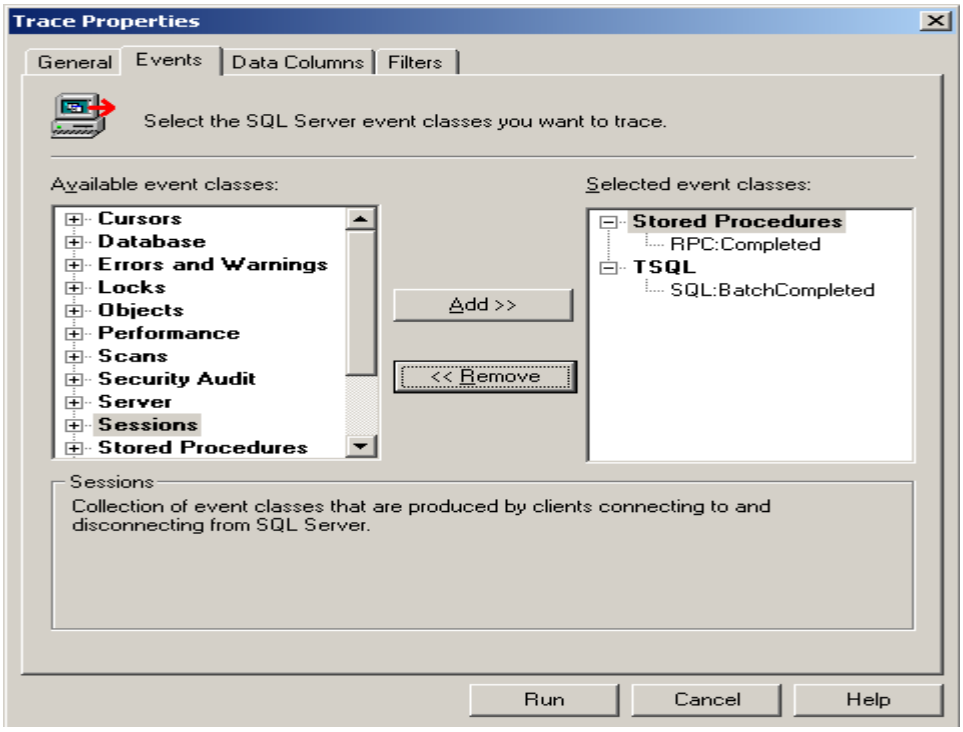


Figura 21: Relação final de eventos que devem ser selecionados.

As colunas e ordem de apresentação dos eventos do template SQLProfilerStandard sofrerão duas alterações: na guia Data Columns ordenaremos as linhas capturadas segundo seu tempo de execução e eliminaremos algumas colunas desnecessárias nesse momento.

5) para ordenar as linhas do Profiler segundo a duração do comando executado, selecione Duration e clique em Up até que essa coluna fique logo abaixo de Groups.
6) As colunas removidas foram: NTUserName, Loginname, CPU, Reads, Writes e ClientProcessId. Veja a **Figura 22**.

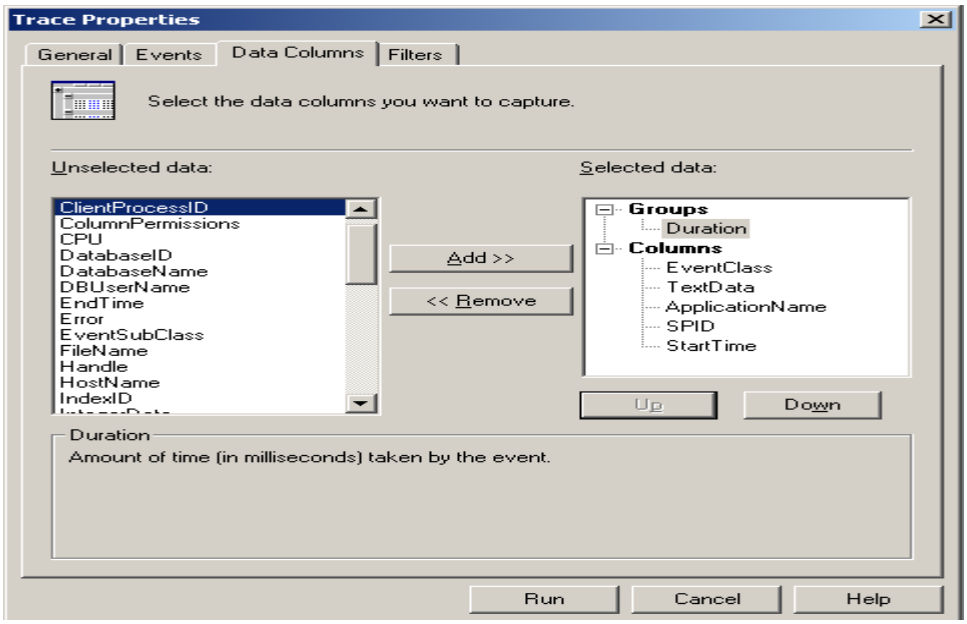


Figura 22: Colunas selecionadas para visualização no Profiler.

Até esse momento, a trace irá capturar todos os batchs e sp´s executados nesse servidor. Seria interessante ligar um filtro de tempo de modo que só fossem capturados os processos com tempo de execução acima de um determinado limite.

7) para conseguir esse efeito, vamos ligar o filtro Duartion na guia Filters, trabalhando com um limite de 2000 milisegundos. Veja a **Figura 23**.

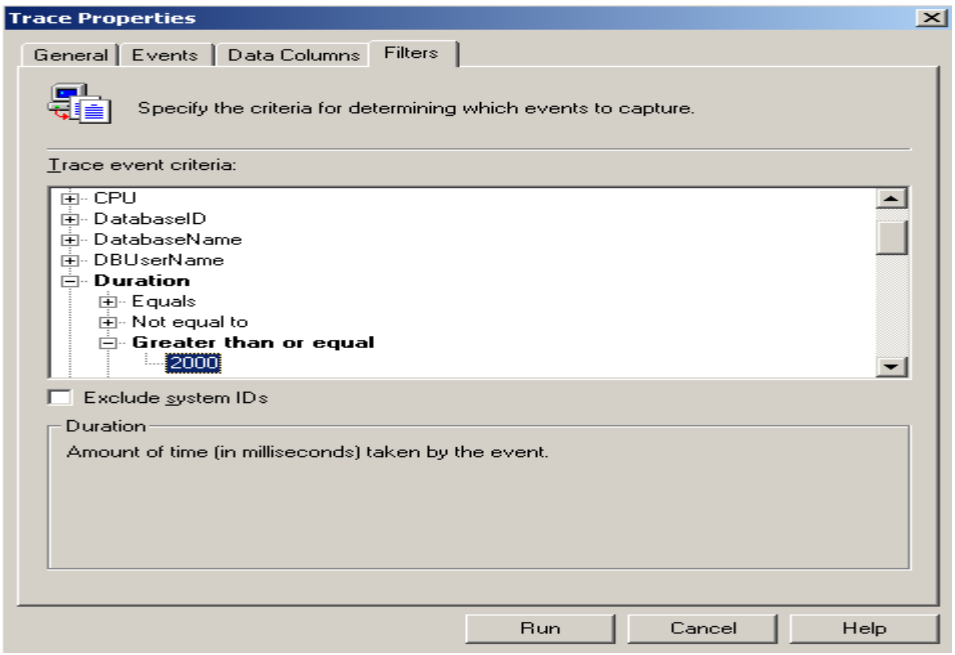


Figura 23: Filtrando comandos T-SQL cujo tempo de duração for superior a 2000 milisegundos.

8) agora inicie a trace clicando <**RUN**>.
9) Para simular uma atividade no SQGBD, abra uma sessão no Query Analyzer, digite as linhas da Listagem 1 e confirme o resultado na tela do Profiler. Veja a **Figura 24**.

```
use Northwind
go

create proc stp_teste
as
  select top1 * from orders
  select top 1 * from [order details]
  waitfor delay '00:00:03'
  select top 1 * from customers
go

exec stp_teste
go
select count(*) from [order details]
go
select top 10 * from orders
go
```

Listagem 1

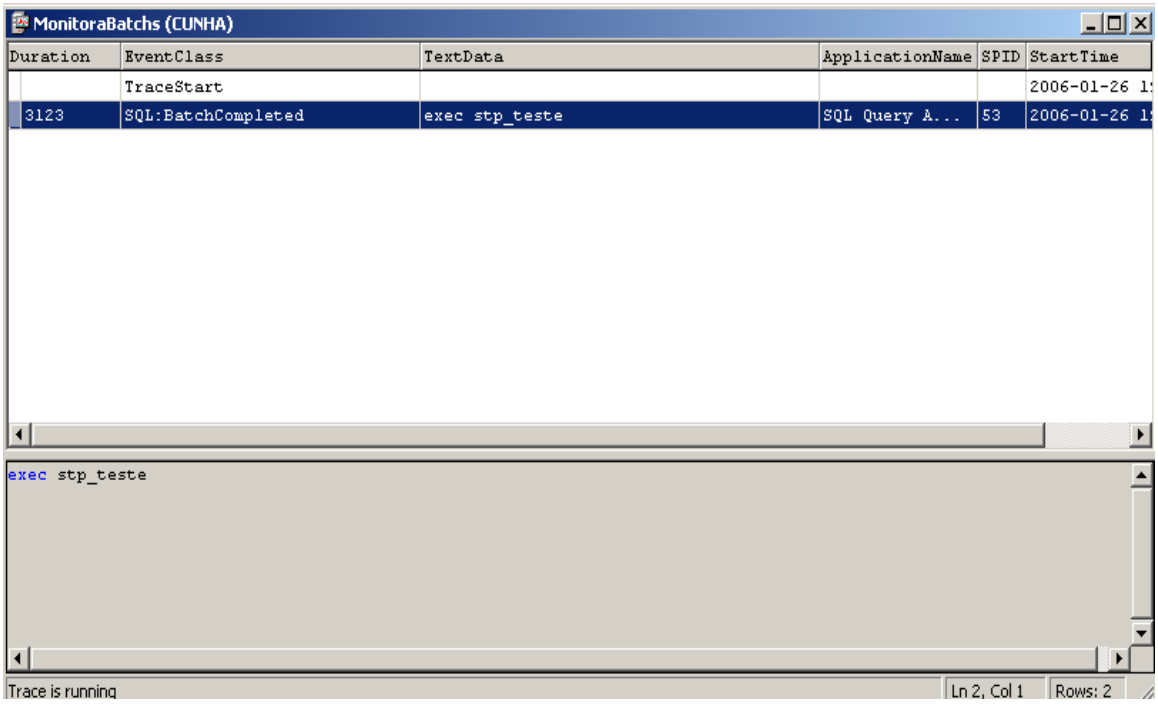


Figura 24 Executando a trace com o filtro Duration ativo.

2. Análise específica: identificando comandos T-SQL com baixa performance.

A trace gerada anteriormente forneceu a relação de batchs e sp´s que estão apresentando tempo de execução superior a 2 segundos. No caso da stored procedute stp_teste, temos que identificar qual comando T-SQL está sendo responsável pela lentidão. Para isto, vamos criar uma trace para analisar somente a sp stp_teste.

1) O ponto de partida é identificar o id do objeto que queremos selecionar. No Query Analyzer, digite:

```
Select object_id (1stp_teste')
-------------------------------
1637580872
```

2) Crie uma trace, selecionando os eventos conforme a **Figura 25**.

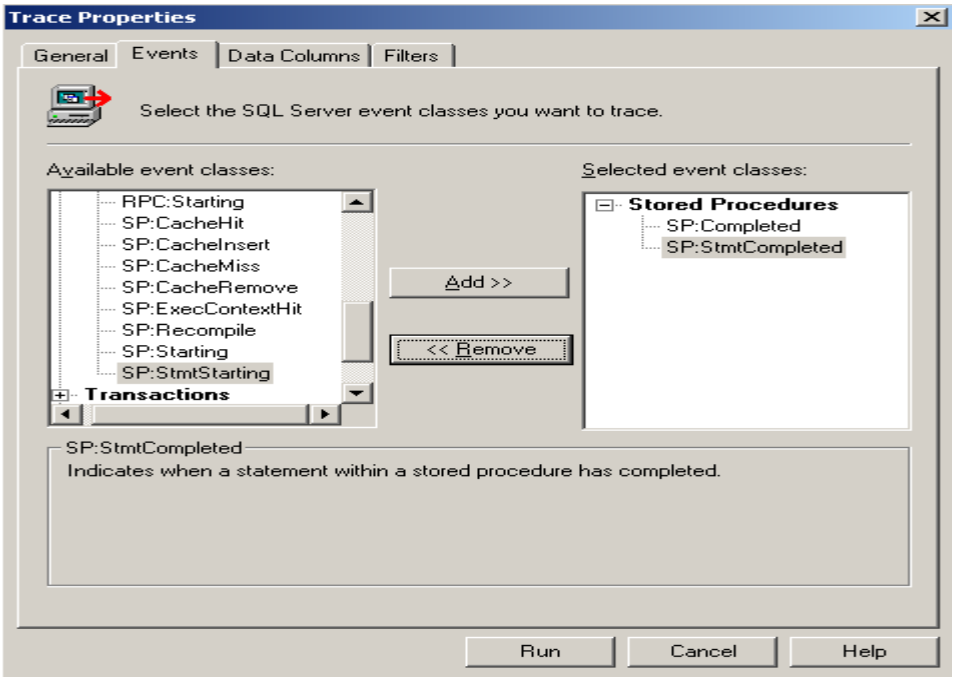


Figura 25: Eventos que devem ser selecionados para visualizar a execução dos comandos T-SQL dentro das sp´s.

SP:Completed – irá gerar uma linha contendo a chamada para a procedure.
SP:StmtCompleted – irá gerar uma linha para cada comando T-SQL executado dentro da sp

3) Na guia Data Columns, selecione os mesmo eventos analisados na **Figura 26.**

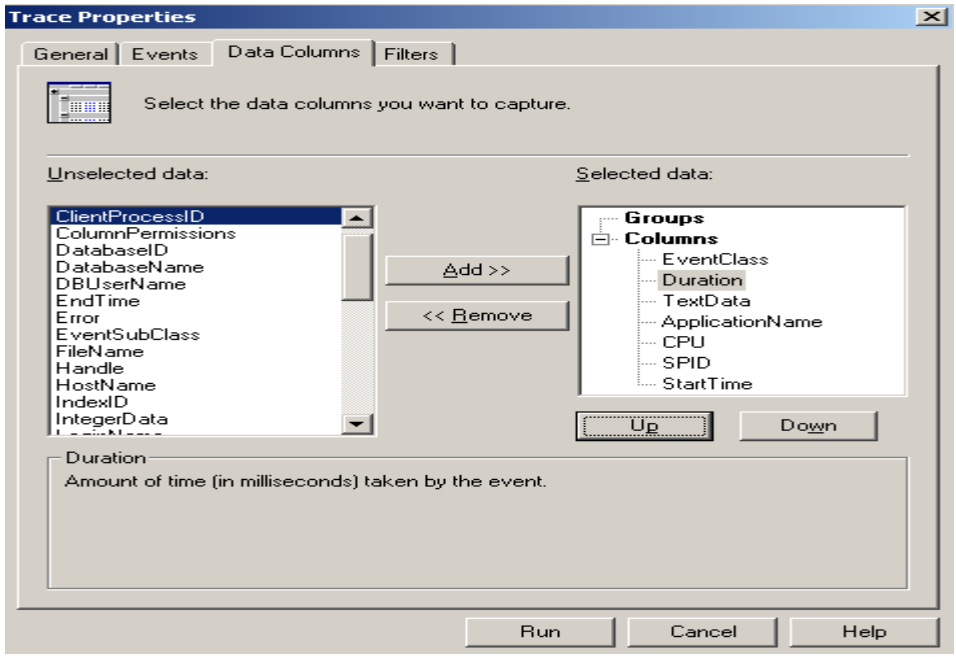


Figura 26: Selecionando colunas para analisar execução de sp.

4) Na guia Filters, siga até ObjectId e informe o id da procedure stp_teste obtida anteriormente. Veja a **Figura 27**

.

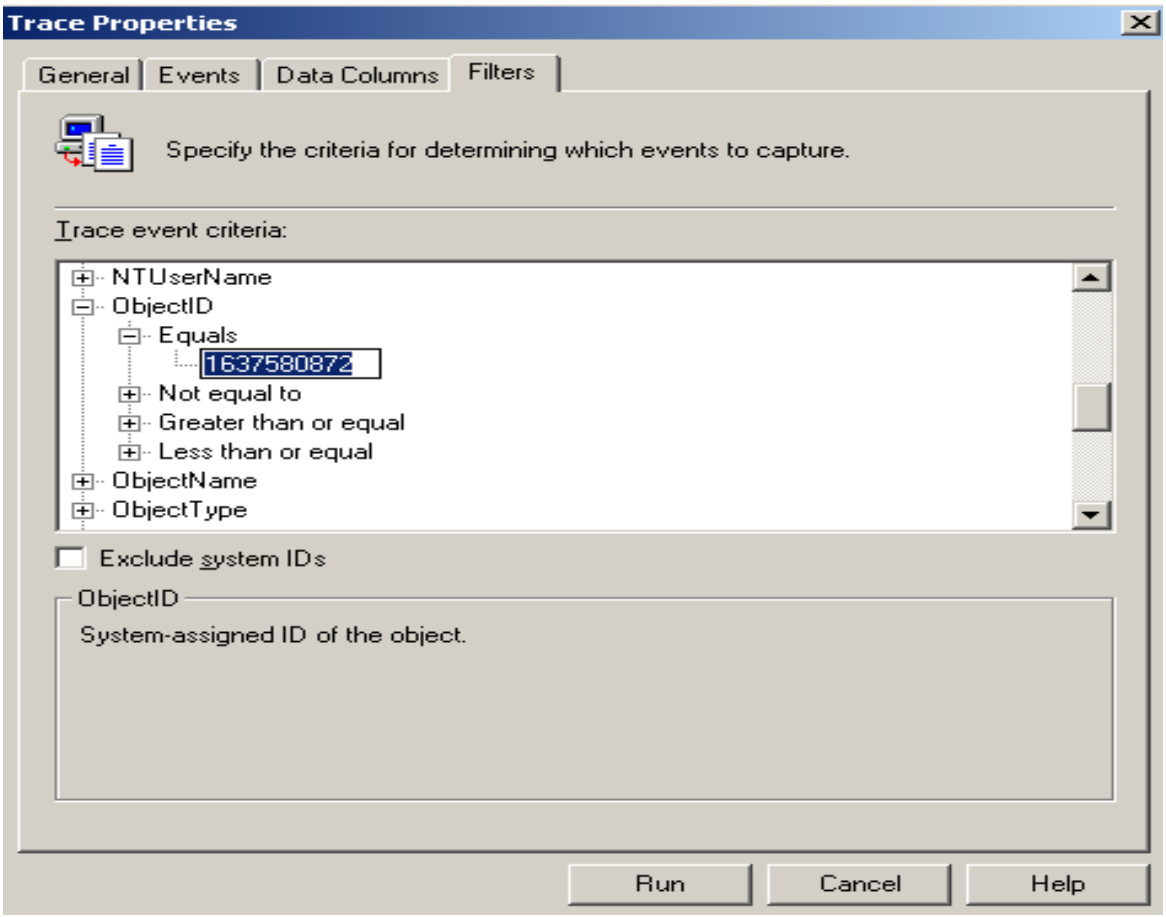
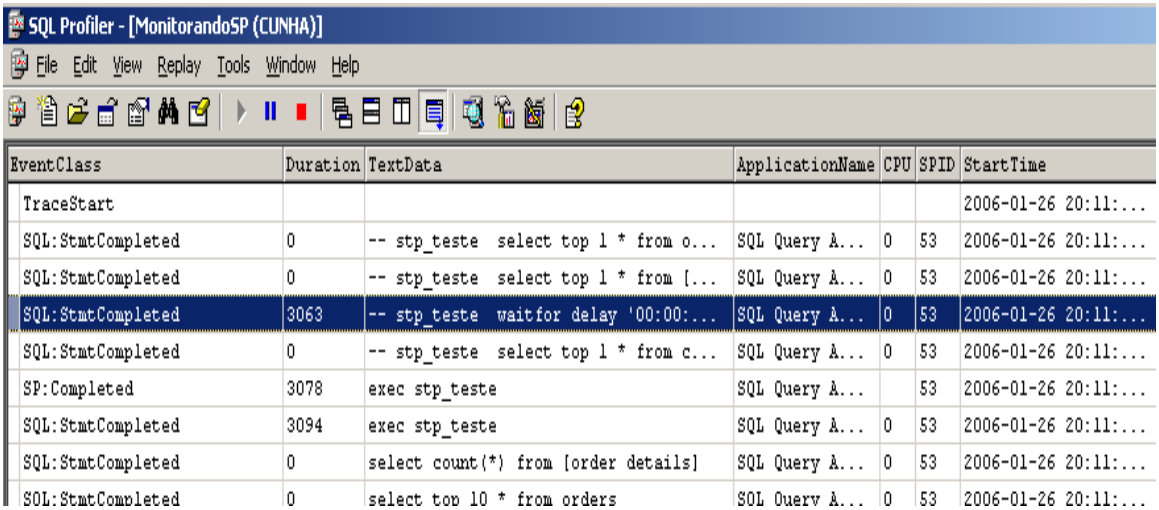


Figura 27: Ligando o filtro ObjectId para filtrar os comandos T-SQL executados na procedure stp_teste.

5) Rode a trace <RUN>
6) Execute a procedure stp_teste. Veja o resultado na tela profiler (**Figura 28**).



EventClass	Duration	TextData	ApplicationName	CPU	SPID	StartTime
TraceStart						2006-01-26 20:11:...
SQL:StmtCompleted	0	-- stp_teste select top 1 * from o...	SQL Query A...	0	53	2006-01-26 20:11:...
SQL:StmtCompleted	0	-- stp_teste select top 1 * from [...	SQL Query A...	0	53	2006-01-26 20:11:...
SQL:StmtCompleted	3063	-- stp_teste waitfor delay '00:00:...	SQL Query A...	0	53	2006-01-26 20:11:...
SQL:StmtCompleted	0	-- stp_teste select top 1 * from c...	SQL Query A...	0	53	2006-01-26 20:11:...
SP:Completed	3078	exec stp_teste	SQL Query A...		53	2006-01-26 20:11:...
SQL:StmtCompleted	3094	exec stp_teste	SQL Query A...	0	53	2006-01-26 20:11:...
SQL:StmtCompleted	0	select count(*) from [order details]	SQL Query A...	0	53	2006-01-26 20:11:...
SQL:StmtCompleted	0	select top 10 * from orders	SQL Query A...	0	53	2006-01-26 20:11:...

Figura 28: Visualização dos comandos executados na procedure stp_teste.

3) Rastreando processos envolvidos em Deadlocks

Deadlock é uma causa freqüente de má performance, pois requer que a conexão escolhida como vítima – e cuja execução de comandos foi abortada- envie novamente o comando para execução. O profiler possui dois eventos que nos ajudam a rastrear deadlocks:

- Lock:Deadlock – informa a ocorrência do erro#1205 associado ao deadlock.
- Lock:DeadlockChain – irá apontar o spid das conexões envolvidas no deadlock.

4) Evitando CacheMiss

Quando acionamos uma sp, seu plano de execução fica em memória. Nesse plano são armazenadas instruções de como a query deverá ser executada: que índice utilizar, o tipo de join selecionado, etc. antes de criar um plano de execução novo, o otimizador faz uma busca na área de cachê destinada a procedures procurando por um plano pré-existente. Se a busca for bem sucedida, o será aproveitado e a sp será executada de acordo com ele.

Existem algumas situações onde o otimizador é obrigado a fazer várias tentativas até encontrar o código pré-compilado. Cada uma dessas tentativas mal sucedidas dispara um evento conhecido por ***SP:CacheMiss***, que pode ser evitado seguindo-se as dicas abaixo:

- Não utilize o prefixo "sp_" para nomear suas sp´s. Quando o otimizador encontra uma procedure com o prefixo sp, sua execução é desencadeada no banco de dados máster. Como a sp não existe nesse banco de dados, seu plano de execução não é encontrado na primeira tentativa, disparando um evento **SP:cacheMiss**. O processo de execução prossegue procurando a sp no banco de dados local, onde o código compilado é encontrado e a sp executada.
- Execute stored procedure qualificando seu dono (owner). Lembre-se que você pode possuir objetos com o mesmo nome, mas com owner diferentes. Troque exec stp_teste por exec dbo.stp_teste.

Principais comandos utilizados na manipulação de traces:

1) Para gerar o script de uma trace: na barra de ferramentas do profiler, selcione File...Save As SQL Script.
2) Verificando as traces ativas no servidor:

Select * from ::fn_trace_getInfo(default)

3) Pasa parar uma trace:

Sp_trace_setstatus 1,0

<1>: id da trace, levantado no item -2
<<0>: parâmetro para stopar a trace.

4) Para iniciar (ou restaurar) uma trace:

St_trace_setstatus 1,1

<1>: id da trace, levantado no item -2
<1>: parâmetro para iniciar a trace.

5) Para encerrar uma trace, ecluindo sua definição da memória do servidor:

St_trace_setstatus 1,2

<1>: id da trace, levantado no item -2
<2>: parâmetro para encerrar a trace.

PS: é necessário stopar antes.

Nota: Funções internas no SQL Server 2000 que possuem o prefixo "fn_" precisam ser chamadas com a adição de "::" (duas vezes o sinal de pontuação dois pontos).

Utilizando filegroups para ganho de performance e gerenciamento de espaço

Filegroups são estruturas lógicas que sustentam os arquivos de dados em um database. Um database padrão possui um arquivo de dados e um arquivo de log; o arquivo de dados está associado a um filegroup chamado PRIMARY. Pode-se criar outros arquivos de dados assim como outros filegroups, mas porque, como e quando criar outros filegroups?

Arquitetura de um database

Apesar do nome singular, um database é uma estrutura formada por pelo menos dois arquivos: um para armazenamento de dados (Máster Data File, extensão .MDF) e outro reservado para o log de transações (Log Data File, extensão .LDF). Veja a **Figura 29**.

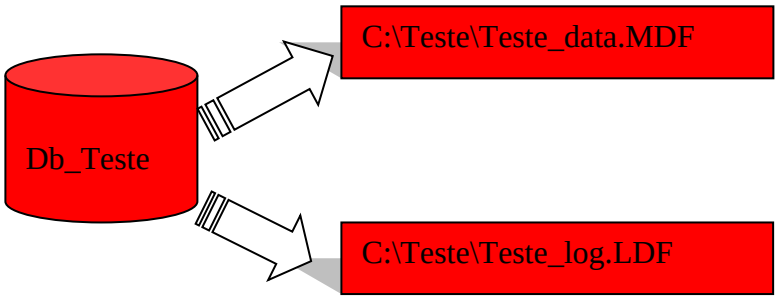


Figura 29: Estrutura típica de um database.

além dos arquivos .MDF e .LDF, é possível criar outros arquivos para armazenamento de dados. Estes arquivos secundários possuem extensão .NDF, de Secondary Data Files e podem ser criados no mesmo filegroup do arquivo .MDF (PRIMARY) ou em outro filegroup. A decisão de utilizar o mesmo filegroup ou criar um novo depende da finalidade do arquivo secundário. A seguir estão listados algumas situações cuja resolução baseia-se na implementação de filegroups:

- O arquivo de dados principal atingiu um tamanho que extrapola a capacidade da unidade de fita DLT utilizada no backup. Esse problema pode ser resolvido com a realocação de tabelas em outro filegroup, distribuindo o backup final em partes menores que não ultraapssem a capacidade da fita. Nesse caso, deve-se criar outro filegroup para armazenar o arquivo secundário.
- Você precisa criar um database com tamanho inicial de 35GB, mas não possui esse espaço em uma única unidade de disco. A distribuição é a seguinte: 20GB na unidade C e 15GB na unidade D. qual a solução? Crie um Máster Data File com 15GB em C e um Secondary Data File com 20GB em D. os dois arquivos constituirão uma unidade única de gerenciamento de espaço. O arquivo secundário criado na unidade D será visto pelo SQL Server como uma extensão natural do arquivo primário. Nesses casos, o arquivo secundário deverá ser criado no mesmo filegroup do arquivo que se deseja expandir (PRIMARY).

- Você pode melhorar a performance de um database criando índices e tabelas em filegroups distintos, localizados em unidades e controladores específicos. Se as páginas de dados das tabelas forem armazenadas em unidades diferentes daquela utilizada para os índices (por exemplo C e D), pesquisas que utilizam índices serão beneficiadas por leituras executadas em paralelo. Nesse caso, deve-se criar outro filegroup para armazenar o arquivo secundário.

A **Figura** 30 é um retrato de um database que utiliza um filegroup secundário para armazenamento de índices.

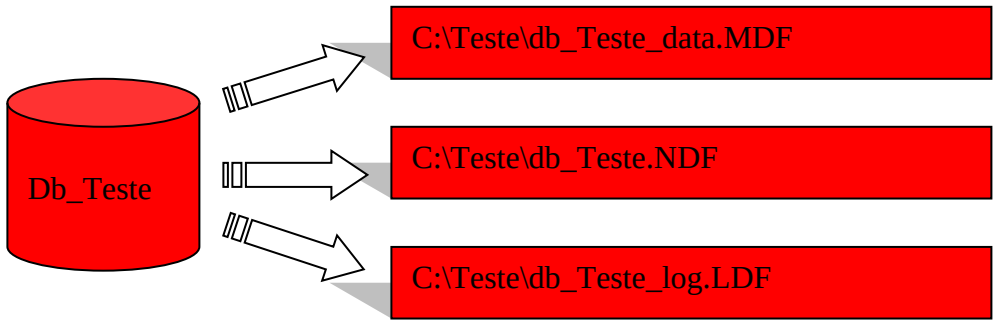


Figura 30: Database que utiliza arquivo secundário (.NDF) para armazenamento de índices

Criando um database através de comandos T-SQL

O comando T-SQL create database pode ser utilizado na criação do database como uma opção ao uso do Enterprise Manager. Veja o comando a seguir:

```
CREATE DATABASE [db_Teste] ON
(
NAME            = N'db_Teste_Data',
FILENAME        = N'c:\Teste\db_Teste_Data.MDF',
SIZE            = 1,
FILEGROWTH      = 10%
)
LOG ON
(
NAME            = N'db_Teste_Log',
```

```
FILENAME        = N'c:\Teste\db_Teste_Log.LDF',
SIZE            = 1,
FILEGROWTH      = 10%
)
```

No exemplo acima, criamos o database db_Teste_data no filegroup PRIMARY representado por c:\Teste\db_Teste_data.MDF. o database também possui um arquivo de Log em c:\Teste\db_Teste_Log.LDF, mas ainda não criamos arquivos secundários.

Criando arquivo secundário

Criando um arquivo secundário através do T-SQL.

```
ALTER DATABASE db_Teste_Data
        ADD FILE
(...
FILENAME = N'c:\Teste\db_Teste_Data2.NDF',
SIZE = 1,
...)
FILEGROWTH = 10%
) TO FILEGROUP [PRIMARY]
```

Comandos DBCC – Database Consistency Checker

No SQL Server 2000 e 2005, através da linguagem **T_SQL**, temos uma série de comandos para manutenção e otimização de tabelas e de índices, comandos estes conhecidos como "comandos **DBCC**". Este grupo de comandos é conhecido como comandos DBCC, porque todos iniciam com o prefixo DBCC. A grande maioria destes comandos é utilizada para verificação da consistência física e lógica de um banco de dados e de seus elementos tais como tabelas e índices. Em muitas situações, o comando, além de fazer a verificação, é capaz de corrigir os problemas encontrados.

Podemos dividir os comandos DBCC em quatro categorias: *manutenção*, *status*, *validação* e *diversos*.

Principais comandos DBCC de Manutenção

Comando DBCC DBREINDEX

Utilizamos este comando para reconstruir um ou mais índices em uma tabela de um Banco de Dados.

Sintaxe conforme Books Online:

```
DBCC DBREINDEX
( [ 'database.owner.table_name'
  [, index_name
  [, fillfactor ]
]
]
) [ WITH NO_INFOMSGS ]
```

Algumas observações a respeito deste comando:

- Não podemos utilizar este comando em tabelas do sistema (master, msdb, etc.)
- Por padrão, somente a role de servidor sysadmin e as roles de banco de Dados db_owner e db_dbaldmin têm permissão para executar este comando.

Vamos a alguns exemplos práticos:

Reconstruir o índice UPKCL_auidind, da tabela authors do banco de Dados pubs.

```
Use pubs
DBCC DBREINDEX ('authors', UPKCL_auidind, 80)
```

O terceiro parâmetro é a definição para Fill Factor, é uma medida para o percentual de espaço a ser deixado em branco, nas páginas do banco de Dados, quando da construção do índice.

Para reconstruir todos os índices de uma tabela, basta não especificar um nome para o índice; apenas coloque dois apóstrofos, conforme indicado no exemplo a seguir, onde são reconstruídos todos os índices da tabela titles do banco de dados pubs:

```
Use pubs
DBCC DBREINDEX ('titles', '', 80)
```

Commando DBCC INDEXDEFRAG

Utilizamos este comando para desfragmentar Clustered e Secondary Indexes de uma tabela.

Sintaxe conforme Books Online:

DBCC INDEXDEFRAG
({ *database_name* | *database_id* | 0 }
, { *table_name* | *table_id* | '*view_name*' | *view_id* }
, { *index_name* | *index_id* }
) [WITH NO_INFOMSGS]

Vamos fazer algumas considerações a respeito deste comando:

- Não podemos utilizar este comando em tabelas do sistema (master, msdb, etc).
- Por padrão, somente a role de servidor sysadmin e as roles de banco de Dados db_owner e db_dbaldmin têm permissão para executar este comando.

- Este comando, além de desfragmentar os índices, compacta suas páginas, levando em conta o valor original do parâmetro **FILL FACTOR**, quando da criação do índice.

Vamos a um exemplo prático:

Desfragmentar o índice UPKCL_auidind, da tabela authors do banco de dados pubs:

Use Pubs
DBCC INDEXDEFRAG (pubs, authors, UPKCL_auidind)

Commando DBCC SHRINKDATABASE

Este comando é utilizado para que possamos reduzir o tamanho de um ou mais arquivos de dados de um Banco de Dados.

Sintaxe conforme Books Online:

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

Algumas observações a respeito deste comando:

- Não podemos reduzir o tamanho de um Banco de Dados a menos do que o tamanho do Banco de Dados model.
- Por padrão, somente a role de servidor sysadmin e a role de Banco de Dados db_owner têm permissão para executar este comando.
- Este comando não irá reduzir um arquivo de Banco de Dados a um tamanho menor do que o tamanho de seus dados.

Vamos a alguns exemplos práticos:

Reduzir o tamanho dos arquivos do Banco de Dados Exemplo1, mantendo um espaço livre de 25% em cada arquivo.

```
Use Exemplo1
GO
DBCC SHRINKDATABASE (Exemplo1, 25)
```

O segundo parâmetro 25 indica o percentual de espaço livre que deve ser mantido, em cada arquivo de dados, após a execução do comando. Por exemplo, um arquivo de dados possui 20 MB, dos quais 10 MB estão ocupados com dados. Após a execução do comando, serão mantidos, evidentemente, os 10 MB de dados, mais 2,5 MB (25%) de espaço livre. Na verdade o SQL Server irá arredondar para 13 MB.

Comando DBCC SHRINKFILE

Utilizamos este comando para reduzir o tamanho de um arquivo de dados (primário ou secundário), ou de um arquivo de log do Banco de dados.

Sintaxe conforme Books Online:

```
DBCC SHRINKFILE
   ( { file_name | file_id }
      { [ , target_size ]
         | [ , { EMPTYFILE | NOTRUNCATE | TRUNCATEONLY } ]
      }
   )
```

Algumas considerações a respeito deste comando:

- Não podemos reduzir o tamanho de um Banco de Dados a menos do que o tamanho do Banco de Dados model.
- Por padrão, somente a role de servidor sysadmin e a role de Banco de Dados db_owner têm permissão para executar este comando.
- Este comando não irá reduzir um arquivo de Banco de Dados a um tamanho menor do que o tamanho de seus dados.

Vamos a alguns exemplos práticos:

Reduzir o tamanho do arquivo primário de dados, do Banco de Dados Exemplo1 a 7 MB.

```
Use Exemplo1
DBCC SHRINKFILE ('Exemplo1-prim', 7)
```

A opção **EMPTYFIL** migra todos os dados do arquivo especificado, para outros arquivos de dados no mesmo Filegroup. Novos dados não poderão ser gravados em um arquivo em que a opção EMPTYFILE foi especificada.; com isso podemos excluir o arquivo, utilizando o comando ALTER DATABASE.

```
Use Exemplo1
Go
DBCC SHRINKFILE ('exemplo1-sec', EMPTYFILE)
GO
ALTER DATABASE Exemplo1
REMOVE FILE 'exemplo1-sec'
```

Comando DBCC UPDATEUSAGE

Este comando informa e corrige erros nas informações e estatísticas sobre o espaço utilizado em disco. Estes erros podem fazer com que o comando sp_spaceused retorne informações incorretas.

Sintaxe conforme o Books Online:

```
DBCC UPDATEUSAGE
  (    { 'database_name' | 0 }
     [ , { 'table_name' | 'view_name' }
     [ , { index_id | 'index_name' } ] ]
  )
  [ WITH    [ COUNT_ROWS ] [ , NO_INFOMSGS ]
     ]
```

Algumas considerações a respeio deste comando:

- Se não existerem problemas nas informações e estatísticas de uso de espaço em disco, este comando não retornará nenhuma mensagem. Este comando tenta corrigir erros nas seguintes colunas da tabela sysindexes: rows, used, reserved e dpages.
- Por padrão, somente a role de servidor sysadmin e a role de Banco de Dados db_owner têm permissão para executar este comando.

Vamos a um exemplo:

DBCC UPDATEUSAGE ('Northwind')

Principais Comandos DBCC de Status

Comando DBCC SHOWCONTIG

Este comando exibe informações sobre a fragmentação dos dados e dos índices de uma determinada tabela.

Sintaxe conforme Books Online:

```
DBCC SHOWCONTIG
  [    ( { table_name | table_id | view_name | view_id }
         [ , index_name | index_id ]
       )
  ]
  [ WITH { ALL_INDEXES
           | FAST [ , ALL_INDEXES ]
           | TABLERESULTS [ , { ALL_INDEXES } ]
           [ , { FAST | ALL_LEVELS } ]
         }
  ]
```

Algumas observações a respeito deste comando:

- DBCC SHOWCONTIG é utilizado para determinar o quão fragmentada está uma tabela. A fragmentação ocorre devido a operações que alteram dados, como inserção, alteração e exclusões. Esta fragmentação pode prejudicar o desempenho de pesquisas realizadas nos dados da tabela. A queda no desempenho pode ser pior no caso de consultas que utilizam uma ou mais cláusula Join.
- Por padrão, somente a role de servidor sysadmin e as roles de Banco de Dados db_owner e db_ddladmin têm permissão para executar este comando.
- Com o comando DBCC SHOWCONTIG, pode-se utilizar as seguintes opções:
    - **WITH FAST**: determina que seja feita uma verificação rápida nos índices;

- **WITH TABLERESULTS**: exibe o resultado da verificação em forma de tabela;
- **WITH ALL_INDEXES**: efetua a verificação em todos os índices de uma tabela ou view;
- **WITH ALL_LEVELS**: somente pode ser utilizada em conjunto com a opção TABLRESULTS. Retorna informações mais detalhadas para cada nível dos índices.

Vamos a alguns exemplos:

Utilizar o comando DBCC SHOWCONTIG para retornar informações sobre todos os índices de todas as tabelas, do Banco de Dados Northwind.

```
Use Northwind
DBCC SHOWCONTIG WITH TABLERESULTS, ALL_INDEXES
```

Também poderíamos retornar as informações sobre a fragmentação em uma única tabela, conforme o exemplo a seguir:

```
Use Northwind
DBCC SHOWCONTIG (Orders)
```

Commando DBCC USEROPTIONS

Com este comando obtemos informações sobre as opções definidas para conexão ativa com o Banco de Dados.

Sintaxe conforme Books Online:

DBCC USEROPTIONS

Exemplo:

```
DBCC USEROPTIONS
```

Principais Comandos de Validação

Comando DBCC CHECKDB

Faz a verificação da alocação do espaço nas páginas de dados e da integridade estrutural de todos os objetos de um Banco de Dados. Além da verificação, este comando é capaz de reparar problemas com a alocação de espaço no Banco de Dados. Dependendo do tamanho do Banco de Dados e do volume de dados, este comando pode demorar um bom tempo para ser executado.

Sintaxe conforme Books Online:

```
DBCC CHECKDB
   ( 'database_name'
         [ , NOINDEX
           | { REPAIR_ALLOW_DATA_LOSS
             | REPAIR_FAST
             | REPAIR_REBUILD
             } ]
   )   [ WITH { [ ALL_ERRORMSGS ]
              [ , [ NO_INFOMSGS ] ]
              [ , [ TABLOCK ] ]
              [ , [ ESTIMATEONLY ] ]
              [ , [ PHYSICAL_ONLY ] ]
              }
       ]
```

Algumas observações a respeito deste comando:

- Por padrão, somente a role de servidor sysadmin e a role de Banco de Dados db_owner é que têm permissão para executar este comando;
- Este comando faz uma verificação da integridade de todos os elementos de um Banco de Dados.

Vamos a alguns exemplos.

Fazer uma verificação de integridade no banco de Dados Northwind.

Use Northwind
DBCC CHECKDB

Também podemos utilizar algumas opções com o comando DBCC CHECKDB. Por exemplo, a opção NOINDEX define que os não clustered das tabelas criadas pelos usuários não devem ser verificados.

DBCC CHECKDB ('Northwind', NOINDEX)

Comando DBCC CHECKTABLE

Faz a verificação da integridade das páginas de dados, índices e páginas com valores de campos do tipo text, ntext e image. Devemos utilizar este comando em tabelas com suspeita de dados corrompidos.

Sintaxe conforme Books Online:

```
DBCC CHECKTABLE
  ( 'table_name' | 'view_name'
    [ , NOINDEX
      | index_id
      | { REPAIR_ALLOW_DATA_LOSS
        | REPAIR_FAST
        | REPAIR_REBUILD }
    ]
  )  [ WITH { [ ALL_ERRORMSGS | NO_INFOMSGS ]
            [ , [ TABLOCK ] ]
            [ , [ ESTIMATEONLY ] ]
            [ , [ PHYSICAL_ONLY ] ]
          }
    ]
```

Algumas observações a respeito deste comando:

- Por padrão, somente a role de servidor sysadmin e a role de Banco de Dados db_owner é que têm permissão para executar este comando;
- É feita uma verificação da integridade física de tabelas.

Vamos a alguns exemplos:

Verificar a integridade da tabela Orders do banco de Dados Northwind.

```
Use Northwind
DBCC CHECKTABLE ('Orders')
```

Verificar a integridade somente das páginas de dados da tabela Orders do Banco de Dados Northwind, isto é, sem fazer a verificação dos índices.

```
Use Northwind
DBCC CHECKTABLE ('Orders') WITH PHYSICAL_ONLY
```

Mais Comandos DBCC

Comando DBCC HELP

Este comando retorna a sintaxe para um determinado comando DBCC.

Sintaxe conforme Books Online:

DBCC HELP **(** '*dbcc_statement*' | **@***dbcc_statement_var* | **'?')**

- Por padrão, somente a role de servidor sysadmin é que tem permissão para executar este comando.

Considere o exemplo:

DBCC HELP ('CHCKDB')

Este comando irá retornar a sintaxe para o comando DBCC CHECKDB.

Agora considere o seguinte exemplo:

DBCC HELP ('?')

Este comando retorna uma listagem de todos os comandos DBCC, sem o prefixo DBCC, para os quais está disponível ajuda, através do comando DBCC HELP.

Nota: para uma referência completa de todos os comandos DBCC, você pode acessar o item DBCC, na referência da linguagem T-SQL, no Books Online.

Definindo opções de bancos de dados

Várias opções de bancos de dados podem ser definidas para cada banco de dados. Apenas o Administrador de Sistema (SA) ou o proprietário do banco de dados pode mudar estas opções. A mudança destas opções só modificará o banco de dados atual; não afetará outros bancos de dados.

As opções de bancos de dados podem serem modificadas com o procedimento armazenado de sistema **sp_dboption**, ou através do Enterprise Manager. O procedimento armazenado **sp_dboption** só afeta o banco de dados atual, mas para modificar opções á nível de servidor, use o procedimento armazenado de sistema **sp_configure**.

Depois de fazer alguma mudança, é emitido automaticamente um checkpoint, de modo que as mudanças são imediatas.

Opções disponíveis

A seguir, temos uma lista das opções mais comuns de banco de dados. Para maiores detalhes em cada uma das opções, veja no Books Online.

As opções marcadas com um asterisco (*) indicam que essa opção pode ser configurada pelo Enterprise Manager; caso contrário, é uma opção só alterável através de procedimentos armazenados.

***ANSI null default**

Controla se o valor padrão para todos os tipos de dados é NULL. A Microsoft põe o padrão em NOT NULL. Se esta opção estiver em TRUE, o padrão será NULL para o banco de dados. Quando se entrar com o comando CREATE TABLE, a não ser que o criador indique explicitamente NOT NULL, a regra se aplicará também à criação da tabela.

ANSI Nulls

Quando em TRUE, as comparações de NULL com qualquer valor vão retornar um NULL. Quando em FALSE, apenas comparações de valores não-Unicode retornarão TRUE se e somente se ambos valores forem nulos. O padrão para essa opção é FALSE.

ANSI Warnings

Quando em TRUE, avisos de erro são exibidos, quando ocorrerem condições tais como divisão por zero ou valores nulos aparecerem em funções de agregação. Por padrão, é FALSE.

***autoclose**

Quando em TRUE, o banco de dados é fechado automaticamente quando o último usuário encerra a conexão. Isto é muito útil para ambientes pequenos, mas deve ser evitado nos casos em que conexões são constantemente feitas e encerradas. A quantidade de carga adicional gerada pela abertura e fechamento de um banco de dados podem ter efeitos negativos em um ambiente de produção.

autoshrink

Quando em TRUE, o SQL Server periodicamente reduzirá os arquivos do banco de dados se necessário.

***dbo use only**

Quando em TRUE, apenas o dbo (proprietário do banco de dados) tem acesso ao banco de dados. Use esta opção quando estiver executando reparos em bancos de dados.

published

Utilizado para relicação, quando *published* estiver em TRUE, indica que a publicação está habilitada. Colocar essa opção em FALSE desabilita a publicação.

***read only**

Se TRUE indica que o banco de dados é somente para leitura. FALSE permite acesso para leitura/escrita.

***recursive triggers**

Quando TRUE, é permitido o disparo de gatilhos recursivos [recursive triggers]. Quando FALSE (o padrão), gatilhos não podem disparar recursivamente. Um gatilho recursivo é aquele que dispara na tabela que o originou, causando uma atualização em outra tabela, a qual causa uma atualização na tabela que originou o gatilho.

***select into / bulk copy**

Permite que o banco de dados aceite ações não registradas em log, tais como SELECT INTO e o utilitário BCP fazem.

***single user**

Permite que apenas um usuário acesse o banco de dados.

subscribed

Quando em TRUE, o banco de dados pode ser assinado para publicação.

***torn page detection**

Se TRUE, o SQL Server detectará leituras incompletas em disco, e fará com que sejam marcadas. Quedas de energia ou outros defeitos podem causar essas leituras incompletas.

Truncate log on Checkpoint (*trunc. Log on chkpt.)

Quando estiver em TRUE, o SQL Server trunca o log de transações toda vez que encontrar um checkpoint. Esta opção é usada freqüentemente para desenvolvimento, fazendo com que o log de transações não fique cheio com tanta freqüência. Você não deve utilizar esta opção em um sistema "real".

Definindo opções do banco de dados com sp_dboption

Para mudar as opções de um banco de dados com o procedimento armazenado sp_dboption, faça o seguinte:

Sintaxe:

```
sp_dboption ['banco_de_dados'] [,'opção'] [,'valor']
```

Por exemplo:

```
sp_dboption 'pubs', 'read only', 'true'
```

Para ver o estado atual das opções do banco de dados **pubs**, entre com o seguinte comando:

```
sp_dboption 'pubs'
```

Todas as opções que estiverem ativadas são listadas.

Definindo opções do banco de dados pelo Enterprise Manager

Quando se utiliza o Enterprise Manager para configurar as opções do banco de dados, você só tem acesso a um subconjunto (cerca de metade) das opções realmente disponíveis.

Para mudar opções do banco de dados com o Enterprise Manager, faça assim:

1. Expanda o grupo do servidor.

2. Expanda o servidor.
3. Expanda os bancos de dados.
4. Clique com o botão direito no banco de dados que você quer mudar, e então clique em Propriedades [Porperties].
5. Selecione as opções a mudar.
6. Clique em OK quando tiver acabado.

Verificando propriedades do banco de dados

A seguir você vê alguns procedimentos armazenados de sistema, freqüentemente utilizados, que exibem informações sobre bancos de dados e opções de bancos de dados.

- **sp_dboption**: como visto acima, mostra todas as opções disponíveis para o banco de dados em que se estiver posicionado.
- **sp_helpdb**: informações sobre todos bancos de dados em um servidor. Fornece nome do banco de dados, tamanho, proprietário, ID, data de criação, e opções.
- **sp_helpdb** *nome_banco_de_dados*: informações sobre um banco de dados específico apenas. Fornece nome do banco de dados, tamanho, proprietário, ID, data de criação, e opções. Além disso, lista os arquivos para dados e log de transações.
- **sp_spaceused** [*nome_objeto*]: resumo do espaço de armazenamento que um banco de dados, log de transações, ou objeto de banco de dados utiliza.

Considerações para melhor gerenciamento

Para que você possa trabalhar com mais tranqüilidade e eficiência com bancos de dados, considere os seguintes fatos.

- Para obter melhor desempenho e segurança, armazene o banco de dados e o log de transações em discos físicos separados.
- Desabilite o cache de escrita nos controladores de disco, a menos que o mecanismo de cache de escrita seja especificamente projetado para servidores de bancos de dados.
- Faça backup do banco de dados master imediatamente depois de criar ou modificar bancos de dados. Em geral, é uma boa idéia fazer backup dos bancos de dados regularmente.
- Garanta que você tenha espaço suficiente para o log de transações. Se você ficar sem espaço, você não será capaz de modificar ou acessar seu banco de dados. Para evitar ficar sem espaço, faça o seguinte:
    - Aloque espaço suficiente para acomodar o crescimento.
    - Monitore freqüentemente o espaço total sendo usado.
    - Use a opção de crescimento automático para aumentar o espaço em disco automaticamente.
    - Configure um alerta para te avisar quando o espaço disponível no log de transações esteja abaixo de 25 por cento do espaço total do log de transações.

Segurança

Conceitos

Criando logins do SQL Server

Criando usuários do banco de dados

Criando grupos de usuários

Definindo permissões

Objetivos:
- Conhecer os recursos do SQL Server para controle de acesso ao banco de dados;
- Aprender a criar logins de usuário e usuários do banco de dados.

Conceitos

Os recursos de segurança do SQL Server permitem determinar:

- Quais usuários podem usar o SQL Server.
- Quais usuários podem acessar cada banco de dados.
- As permissões de acesso para cada objeto de banco de dados e para cada usuário.
- As permissões de acesso para cada comando SQL em cada banco de dados, para cada usuário.

Existem quatro barreiras para que os usuários possam acessar dados em um servidor SQL Server:

- O sistema operacional de rede; o usuário deve efetuar logon na rede.
- A autenticação do SQL Server; o usuário deve ter uma conta no SQL Server.
- A autenticação de banco de dados; o ID do usuário deve existir em uma tabela de sistema do banco de dados (mais especificamente, a tabela *sysusers*)
- A autenticação de objetos; o usuário deve ter permissões para acessar qualquer objeto (tabelas, visões, entre outros).

Autenticação de usuários

Quando um usuário tenta acessar um servidor SQL Server, ele pode ser autenticado de duas maneiras: pela Autenticação do Windows NT ou pela Autenticação do SQL Server.

Não confunda isso com modo de segurança, que é um tópico muito semelhante. A autenticação do Windows NT se aproveita da segurança embutida no Windows NT Server, a qual inclui características como senhas criptografadas, senhas que expiram, tamanho mínimo de senhas, bloqueio de conta, e restrição de acesso com base em nomes de computador.

O SQL Server pode confiar no Windows NT para autenticar logins, ou pode ele mesmo autenticar os logins.

Quando o Windows NT autentica o login, o SQL Server processa o login assim:

- Quando um usuário se conecta ao SQL Server, o cliente abre uma conexão confiável com o SQL Server, na qual são passadas as contas de usuário e de grupo do cliente para o SQL Server.
  Uma *conexão confiável* [trusted connection] é uma conexão de rede com o SQL Server que consegue ser autenticada pelo Windows NT. Para ocorrer uma conexão confiável, as bibliotecas de rede [net-libraries] Named Pipes ou Multiprotocol devem estar sendo utilizadas tanto pelo cliente quanto pelo servidor SQL Server. Caso a biblioteca de rede sendo utilizada pelo cliente ou pelo servidor não seja uma dessas duas, a conexão de rede é *não-confiável* e a autenticação do Windows NT não pode ser utilizada.
- Se o SQL Server encontra a conta de usuário ou de grupo na lista de contas de login do SQL Server, na tabela de sistema *syslogins*, ele aceita a conexão.
  O SQL Server não precisa de revalidar uma senha, já que o Windows NT já a validou.
- Nesse caso, a conta de login no SQL Server, do usuário, é a conta de usuário ou de grupo do Windows NT, a que tiver sido definida como a conta de login do SQL Server.
- Se vários computadores com servidores SQL Server participam em um domínio ou um grupo de domínios confiáveis, basta efetuar logon em um único domínio para ter acesso a todos os servidores SQL Server.

Nota: O SQL Server não irá reconhecer grupos nem usuários que foram excluídos e depois recriados no Windows NT. Os grupos devem ser excluídos do SQL Server e adicionados novamente, pois o SQL Server usa o identificador de segurança (SID) do Windows NT para identificar um grupo ou usuário. E um grupo ou usuário excluído e depois criado novamente com o mesmo nome no Windows NT, terá um SID diferente.
Quando o SQL Server autentica o login, ocorre o seguinte:

- Quando um usuário se conecta ao SQL Server com um nome de usuário e senha de uma conta do SQL Server, o mesmo verifica que um login existe na tabela de sistema *syslogins* e que a senha especificada é igual a que se tem gravada.

Se o SQL Server não tem uma conta de login com esse nome de usuário ou a senha não é a que se tem gravada, á autenticação falha e a conexão é recusada.

Modos de segurança

Um modo de segurança se refere a como o DBA (administrador do banco de dados) configura o SQL Server para autenticar usuários. Um servidor pode usar um de dois modos

de segurança: Windows NT e mista [mixed]. A diferença entre esses modos de segurança é como a segurança do SQL Server se integra com o Windows NT:

Modo de autenticação mista do SQL Server [SQL Server Mixed Authentication Security Mode]: Nesse nidi de segurança, um usuário pode conectar-se ao SQL Server usando a Autenticação do Windows NT, ou a Autenticação do SQL Server. Ao tentar conectar-se com o SQL Server, verifica-se se você está usando ou não uma conexão confiável. Ocorre então o seguinte:

- Se você estiver usando uma conexão confiável, o SQL Server tentará autenticar o seu login do Windows NT, verificando se o seu nome de usuário tem permissão para conectar-se ao servidor SQL Server. Caso seu nome de usuário não tenha permissão para conectar-se ao SQL Server, lhe será pedido um nome de login e senha.
- Caso você não esteja usando uma conexão confiável, lhe será logo pedido um login e senha.
- Seu login e senha são verificados na tabela de sistema *syslogins*. Se o nome de login for válido e a senha correta, você poderá conectar-se ao servidor SQL Server.

Quando o SQL Server lhe pede um login e senha, ele usa seu próprio cadastro de usuários, independente do banco de dados de contas do Windows NT. Os logins de usuário devem ser cadastrados no SQL Server.

Modo de autenticação de segurança do Windows NT [Windows NT Server Authtentication Security Mode]: Se se opta por usar o modo de segurança do Windows NT, só o mecanismo de autenticação do Windows NT é utilizado para autenticar usuários para o SQL Server. O nome de usuário que foi usado para se conectar à rede NT é o mesmo nome usado para o SQL Server. Esse nome de usuário e a senha não precisam ser informados novamente. Se o usuário for autorizado (ou seja, tiver um registro na tabela de sistema *syslogins*) a conectar-se ao SQL Server, então ele poderá conectar-se. Nesse modo de segurança, só é possível se conectar ao SQL Server através de uma *conexão confiável*. Se esta opção for escolhida, deve-se ter certeza de que todos os clientes estejam rodando em sistemas Windows, e que possam conectar-se ao SQL Server usando uma conexão confiável.

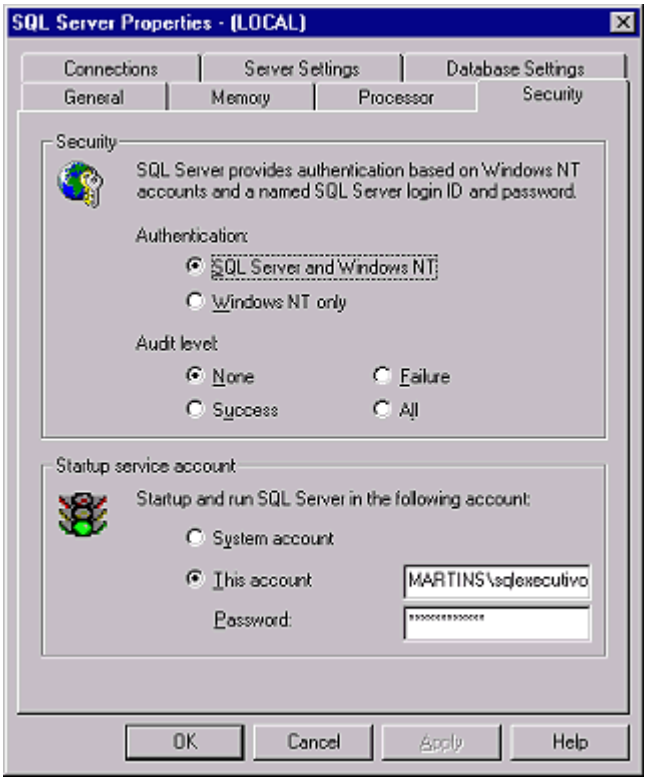
Vantagens de cada um dos modos de segurança

Modo de segurança do Windows NT	**Modo de segurança mista**
Recursos avançados de segurança	Clientes não-Windows e usando browser podem usar esse modo para conectar-se.
Adicionar grupos como uma conta.	Camada adicional de segurança sobre o Windows NT
Acesso rápido.	

Definindo o modo de segurança

Para definir o modo de segurança, você deve fazer o seguinte:

- No Enterprise Manager, selecione o servidor cujo modo de segurança você quer definir.
- Clique com o botão direito e selecione **Properties**.
- Na tela que aparecer, selecione a guia **Security**.



- Em Authentication, caso você selecione "SQL Server and Windows NT", o modo de segurança mista (mixed mode) estará sendo definido.
- Caso você selecione "Windows NT only", o modo de segurança do Windows NT estará sendo definido.
- Em qualquer dos casos, você deve parar e reiniciar o serviço MSSQL Server para que a mudança tenha efeito. Para isso, você pode usar, entre outras ferramentas, o Service Manager.

Logins

Um *login* do SQL Server (ou *login ID*) é um nome que identifica um usuário para o SQL Server. Cada login tem uma senha, que deve ser informada no caso da segurança mista (ver abaixo).

O SQL Server cria automaticamente um login chamado 'sa' (administrador do sistema), que não deve ser excluído. O 'sa' tem permissão para fazer praticamente tudo no banco de dados: criar bancos de dados, tabelas, criar outros logins etc. O sa pode conceder permissões para outros usuários poderem fazer algumas tarefas.

Também é criado automaticamente o login BUILTIN\Administrators. Esse login é a conta padrão de login para todos os administradores do Windows NT. Esse login tem todos os direitos no SQL Server e em todos os bancos de dados.

Nomes de usuário no banco de dados

Se você possui um login, não quer dizer que tenha acesso a todos os bancos de dados. É preciso ter também um *nome de usuário de banco de dados* [database user ID], que é relacionado com o login e permite acesso a um banco de dados específico. O nome de usuário pode ser específico do login.

O usuário que cria um banco de dados é o *dono* do banco de dados [database owner]. Dentro do banco de dados, o dono é conhecido pelo nome especial 'dbo'. Outros usuários podem ter nomes diferentes, geralmente de acordo com o seu login. O dono do banco de dados pode conceder permissões para outros usuários de criar e excluir objetos dentro do banco de dados.

O usuário que cria um objeto (tabela, visão, procedimento etc.) no banco de dados é o *dono* deste objeto. O dono tem inicialmente todas as permissões no objeto criado, mas ele pode conceder essas permissões a outros usuários se desejar.

Um login pode ter um *alias* [apelido] dentro de um banco de dados, que é o nome de outro usuário. Nesse caso, dentro daquele banco de dados, ele funciona como se fosse aquele usuário e tem as mesmas permissões dele. Vários usuários (logins) diferentes podem ter o mesmo alias. Esse é um recurso que existe no SQL Server 7.0, apenas para compatibilidade com versões anteriores, já que através de papéis [roles] e da atribuição de permissões aos papéis, o que era feito usando *aliases*, pode ser feito de maneira muito mais eficaz.

O usuário *guest* [convidado] é um nome especial que existe em todo banco de dados e permite a qualquer login usar o banco de dados, mesmo que não tenha um nome de usuário relacionado.

Papéis [Roles]

Na sua essência, um *papel* [role] é um grupo de usuários que têm necessidades semelhantes de acesso ao SQL Server. Mas, os papéis são um pouco mais complexos do que isso. Por exemplo, há uma porção de tipos diferentes de papéis do SQL Server, incluindo os seguintes:

- Papéis predefinidos de servidor [Predefined server roles]

- Papéis predefinidos de bancos de dados [Predefined database roles]
- O papel público [Public role]
- Papéis personalizados de bancos de dados [Custom database roles]

Papéis de aplicação são um tipo especial de papéis que são atribuídos a uma aplicação específica que foi projetada para acessar os dados do SQL Server. Por exemplo, se um usuário precisa de acessar um tipo específico de dados, ao invés de atribuir permissão explícita ao usuário para acessar os dados, o acesso aos dados é dado ao usuário utilizando a aplicação à qual foi atribuído um papel de aplicação. Isso significa que um usuário apenas terá acesso aos dados usando essa aplicação específica. Papéis de aplicação são atribuídos a aplicações, não a usuários.

Papéis predefinidos de servidor [Predefined Server Roles]

Em versões anteriores do SQL Server era difícil delegar áreas administrativas a outras pessoas. Por exemplo, você poderia querer se designar como o DBA senior, com a habilidade de executar qualquer tarefa no SQL Server, que precisasse ser executada. Além disso, você poderia querer delegar algumas das tarefas administrativas para outros, e ao mesmo tempo restringir exatamente o que eles poderiam fazer. Embora isso fosse possível em versões anteriores do SQL Server, era difícil de implementar. O SQL Server 7.0 solucionou esse problema incluindo o que são chamados de papéis predefinidos de servidor (também conhecidos como papéis fixos de servidor).

O SQL Server inclui um total de sete diferentes papéis predefinidos de servidor, cada um com um conjunto de permissões administrativas diferentes. Isso te permite definir vários ajudantes administrativos, com diferentes níveis de capacidade, para ajudá-lo a administrar o SQL Server. Tudo que você precisa fazer é adicionar o login dos mesmos ao papel desejado. Todos os papéis de servidor são predefinidos pelo SQL Server. Você não pode criar seus próprios papéis de servidor.

Os papéis predefinidos de servidor são definidos ao nível do servidor SQL Server, não ao nível de banco de dados. Isso significa que qualquer um que pertença a um desse papéis predefinidos de servidor tem permissões específicas para gerenciar os servidore SQL Server e todos os bancos de dados gerenciados pelo SQL Server. As tarefas administrativas que cada login pode executar dependem apenas de qual papel predefinido de servidor a que ele pertença. Os papéis predefinidos de servidor são:

- Administradores de sistema [System Administrators] (sysadmin): Este é o mais poderoso de todos os papéis. Qualquer um que pertença a esse papel pode realizar qualquer tarefa no SQL Server, inclusive sobrepor-se a qualquer dos outros papéis predefinidos de servidor. Esse papel é o equivalente à conta SA em versões anteriores do SQL Server (a conta SA, por padrão faz parte desse grupo).
- Criadores de bancos de dados [Database Creators] (dbcreator): Eles têm a habilidade de criar e alterar bancos de dados individuais.
- Administradores de discos [Disk Administrators] (diskadmin): Têm a capacidade de gerenciar arquivos de disco.

- Administradores de processos [Process Administrators] (processadmin): Têm a capacidade de gerenciar os vários processos sendo executados no SQL Server.
- Administradores de segurança [Security Administrators] (securityadmin): Eles têm a capacidade de gerenciar logins para um servidor.
- Administradores de servidor [Sever Administrators] (serveradmin): Têm a capacidade de realizar configurações a nível de servidor.
- Administradores de configuração [Setup Administrators] (setupadmin): Têm a capacidade de instalar a replicação no SQL Server, e gerenciar procedimentos armazenados.

Geralmente, você não precisará de todos esses papéis quando for delegar tarefas administrativas do SQL Server para ajudantes. Em muitos casos, você provavelmente só atribuirá seus ajudantes a um ou dois papéis, dando-lhes as permissões específicas que eles precisam para executar as tarefas que você delegou a eles. Os usuários podem pertencer a mais de um papel ao mesmo tempo.

Papéis predefinidos de bancos de dados [Predefined Database Roles]

Sob vários aspectos, os papéis predefinidos de bancos de dados são semelhantes aos papéis predefinidos de servidor. Papéis predefinidos de bancos de dados atribuem tipos específicos de permissões a para cada um dos nove papéis predefinidos. A principal diferença entre papéis predefinidos de servidor e papéis predefinidos de bancos de dados é que os papéis predefinidos de bancos de dados são específicos para cada banco de dados e não se estendem a vários bancos de dados. Como os papéis predefinidos de servidor, os papéis predefinidos de bancos de dados podem ser usados por você para ajudá-lo a distribuir as tarefas administrativas do SQL Server para outros. Os papéis predefinidos de bancos de dados são:

- Proprietário do banco de dados [Database Owner] (db_owner): Eles têm permissões de propriedades em um banco de dados e podem executar qualquer tarefa de configuração ou manutenção em um banco de dados particular. Eles também podem executar todas as atividades dos outros papéis de bancos de dados, e sobrepor-se a qualquer dos outros papéis. Em versões anteriores do SQL Server, esse papel é muito semelhante ao ID de usuário de banco de dados DBO. (a conta DBO faz parte desse papel em todos os bancos de dados).
- Administrador de acesso do banco de dados [Database Access Administrator] (db_accessadmin): Têm a capacidade de gerenciar IDs de usuário de banco de dados para um banco de dados.
- Leitor de dados do banco de dados [Database Data Reader] (db_datareader): Têm a capacidade de ver quaisquer dados de todas as tabelas em um banco de dados.
- Escritor de dados do banco de dados [Database Data Writer] (db_datawriter): Têm a habilidade de inserir, modificar ou excluir quaisquer dados de todas as tabelas em um banco de dados.
- Administrador da linguagem de definição de dados [Database Data Definition Language Administrator] (db_ddladmin): Podem criar, modificar, ou excluir

quaisquer objetos de um banco de dados (tabelas, visões, procedimentos armazenados...).

- Operador de backup do banco de dados [Database Backup Operator] (db_dumpoperator): Podem realizar backups do banco de dados.
- Negação de leitura no banco de dados Database Deny Data Reader (db_denydatareader): Um papel especial que permite a seus membros mudar o esquema do banco de dados, mas sem poder ver os dados no banco de dados.
- Negação de escrita no banco de dados [Database Deny Data Writer] (db_denydatawriter): Um papel especial que evita que seus membros alterem qualquer dado em um banco de dados.

Como o DBA, você provavelmente não usará a maioria desses papéis predefinidos de bancos de dados. É provável que você precise de apenas alguns de modo a delegar algumas de suas tarefas administrativas para seus ajudantes. E como com os papéis predefinidos d servidor, usuários podem pertencer a mais de um papel ao mesmo tempo.

O papel público [Public Role]

O papel público é semelhante ao grupo público que era usado em versões anteriores do SQL Sever. Quando criado, todo banco de dados tem o papel público por padrão, assim como todo banco de dados tem papéis predefinidos de banco de dados. O que é único nesse papel é que todos IDs de usuário em um banco de dados automaticamente pertencem a este papel. Sob vários aspectos, ele é semelhante ao grupo Todos [Everyone] do Windows NT Server. Você não pode adicionar ou remover usuários deste papel, ou modificá-lo de qualquer maneira. Tudo que pode ser feito é atribuir permissões a ele. Quaisquer permissões atribuídas ao papel público são automaticamente atribuídas a todos IDs de usuário no banco de dados. O papel público é especialmente útil se você quiser atribuir as mesmas permissões para todos os usuários de banco de dados em um banco de dados, ao mesmo tempo.

Papéis personalizados de banco de dados [Custom Database Roles]

Como uma regra geral, você irá querer se aproveitar do máximo de papéis predefinidos possível. Mas você pode encontrar situações onde nenhum dos grupos predefinidos vai de encontro às suas necessidades. Se esse for o caso, o SQL Server permite que você crie seus próprios papéis de banco de dados.

Se você estiver utilizando a autenticação do Windows NT e usa grupos globais do NT Server para gerenciar usuários, você perceberá que você não precisa realmente de criar papéis personalizados de banco de dados, já que você obtém o mesmo efeito com o uso de grupos globais ao invés de agrupar usuários semelhantes. Mas se você não for um administrador do NT Server e não tiver permissão para criar os grupos globais que você precisa, ou se você estiver utilizando a autenticação do SQL Server, você pode não ter outra escolha, a não ser criar os papéis personalizados de bancos de dados para ajudá-lo a gerenciar melhor seus usuários.

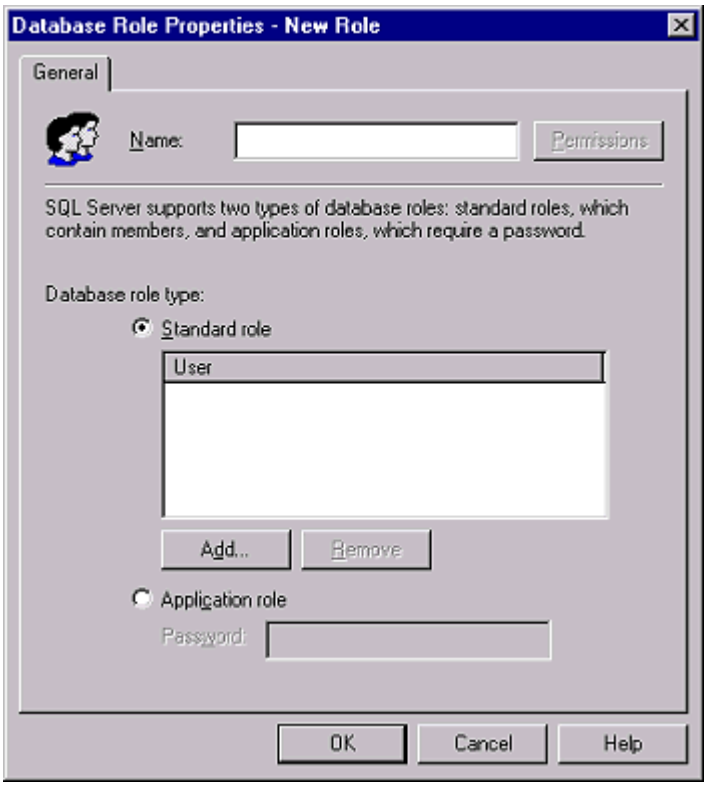
Quando for criar papéis personalizados de banco de dados, tenha o seguinte em mente:

- Papéis personalizados de banco de dados, como ocorre com qualquer papel do SQL Server, são utilizados para agrupar usuários semelhantes que precisam do mesmo conjunto de permissões para acessar o SQL Server.
- Papéis personalizados de bancos de dados são criados dentro de um banco de dados e não podem se estender a vários bancos de dados.
- Usuários podem pertencer a mais de um papel, seja personalizado ou predefinido.
- Papéis personalizados podem incluir usuários do NT Server, grupos globais do NT Server, IDs de usuários de bancos de dados do SQL Server, e outros papéis do SQL Server.

Nota: Se você tiver permissão para a criação de grupos globais e utilizar a autenticação do Windows NT, você deve sempre usar grupos globais ao invés de papéis personalizados de banco de dados. O uso de grupos globais ao invés de papéis personalizados de banco de dados geralmente reduz o tempo necessário para gerenciar as contas de usuário do SQL Server e do NT Server, pois grupos globais funcionam com ambos. Papéis personalizados de banco de dados funcionam apenas no SQL Server. Além disso, grupos globais podem se estender a vários bancos de dados, enquanto papéis são específicos para cada banco de dados, o que os torna menos flexíveis que grupos globais.

Criando e configurando papéis de banco de dados

Veremos agora como criar um papel personalizado de banco de dados. Para isso, n o Enterprise Manager, expanda o banco de dados para o qual você quer criar o papel. Clique em **Roles** com o botão direito e selecione **New Database Role**. Aparece a caixa de diálogo de criação de papéis de banco de dados.



Na caixa "Name" digite o nome do papel de banco de dados que você quer criar. Depois, você deve informar se você está criando um papel padrão [Standard Role] ou um papel de aplicação [Application Role]. Se você escolher criar um papel padrão, você tem a opção de adicionar um ou mais IDs de usuários de banco de dados ao papel agora (clicando em **Add...**). Ou então, você pode pular este passo agora e adicionar IDs de usuários de banco de dados posteriormente, usando as técnicas mostradas em Gerenciando usuários . Se você escolher papel de aplicação, você também dever informar uma senha.
Depois de terminar de informar o que foi pedido, clique em Ok para criar o novo papel de banco de dados. Isso fechará a caixa de diálogo acima e então o novo papel será mostrado no Enterprise Manager.

Nota: Lembre-se que papéis de banco de dados são criados para cada banco de dados. Eles não são compartilhados entre bancos de dados.

Excluindo um papel de banco de dados

Como parte da sua responsabilidade cotidiana de manter o SQL Server, você achará necessário às vezes remover IDs de usuário de bancos de dados e, com menor freqüência, remover papéis de banco de dados que não sejam mais necessários. A remoção de IDs de usuário de banco de dados será vista em Como excluir um ID de um usuário de banco de dados).

Os únicos papéis de banco de dados que podem ser removidos são aqueles que foram criados por você ou por outro DBA. Não é possível remover papéis predefinidos de

banco de dados. Se um papel de banco de dados tem um ou mais IDs de usuário de banco de dados associado a ele, você deve removê-los do papel antes de tentar excluir o papel. Se você tentar excluir um papel sem antes remover os IDs de usuários a ele associados, você verá uma mensagem de erro.

Para excluir um papel de banco de dados, faça o seguinte:

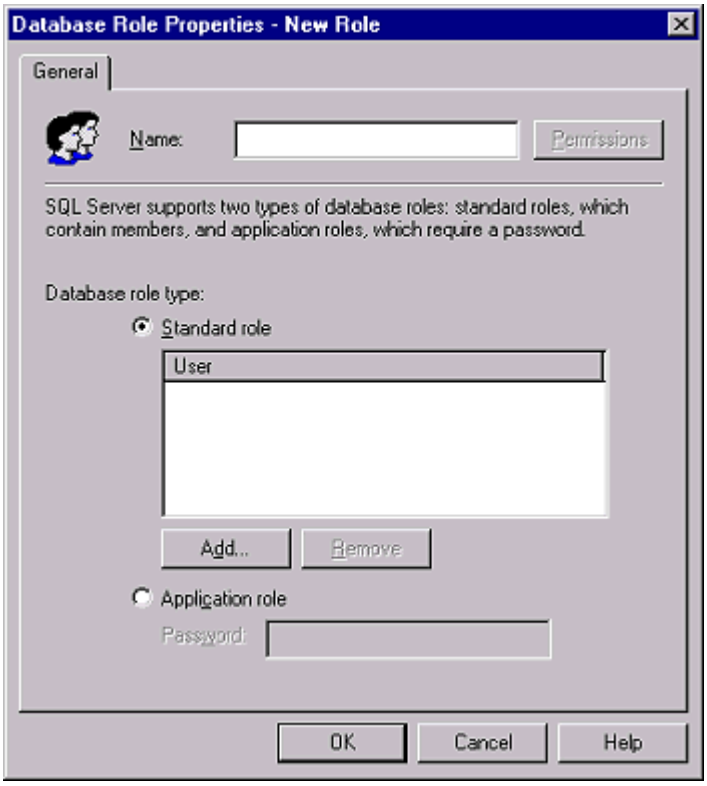
7. No Enterprise Manager, expanda o banco de dados em que está definido o papel que você quer excluir.
8. Clique em Roles e, no lado direito da tela, selecione o papel a ser excluído. Dê um duplo clique no mesmo, e verifique se em baixo de **User**, há algum usuário listado.
9. Caso haja algum usuário, remova-o selecionando-o e clicando no botão **Remove**.
10. Feito isso, clique em Ok, selecione o papel a ser excluído, clique no mesmo com o botão direito e selecione **Delete**.
11. Lhe será perguntado se você de fato quer excluir o papel. Confirme, clicando em **Yes**.

Caso você saiba de antemão que não há usuários associados a esse papel, vá direto para o passo 4.

Configurando um papel de servidor

Como já foi visto, papéis de servidor são usados para atribuir aos logins vários níveis de privilégios administrativos no SQL Server. Você pode atribuir um login a um papel de servidor quando você cria um login (como visto em Gerenciando usuários), ou você pode fazer como será descrito aqui.

Os papéis de servidor vêm embutidos no SQL Server. Novos papéis de servidor não podem ser criados nem os existentes podem ser deletados. Sua única opção ao configurar um papel de servidor é adicionar ou remover logins do papel de servidor em questão.



Para adicionar ou remover um login de um papel de servidor, faça o seguinte:

- No Enterprise Manager, selecione o servidor SQL Server cujos papéis você quer configurar. Expanda-o e abra a pasta **Security**.
- Clique em **Server Roles**. No lado direito da tela aparecem os papéis de servidor. Selecione o papel ao qual você quer adicionar algum login.
- Clique no mesmo com o botão direito e selecione Properties. Aparece a janela abaixo:

- Para adicionar um login ao papel de servidor, clique no botão **Add**. Aparece a caixa de diálogo "Adicionar Membros" [Add Mebers], com todos os logins definidos para o servidor.
- Escolha um ou mais logins para adicionar a esse papel de servidor. Cada vez que você selecionar um login, ele ficará marcado, e assim ficará até que você o clique de novo. Depois de selecionados todos os logins que você quer adicionados ao papel de servidor, clique em Ok. Então você volta para a caixa de diálogo de propriedades do papel de servidor (mostrada acima).
- Caso você queira remover algum login que faz parte de um papel de servidor, selecione-o, na caixa de diálogo de propriedades do papel de servidor, e clique no botão **Remove**.
- Quando você tiver adicionado e/ou removido todos os logins desejados a esse papel de servidor, clique em Ok para concluir.

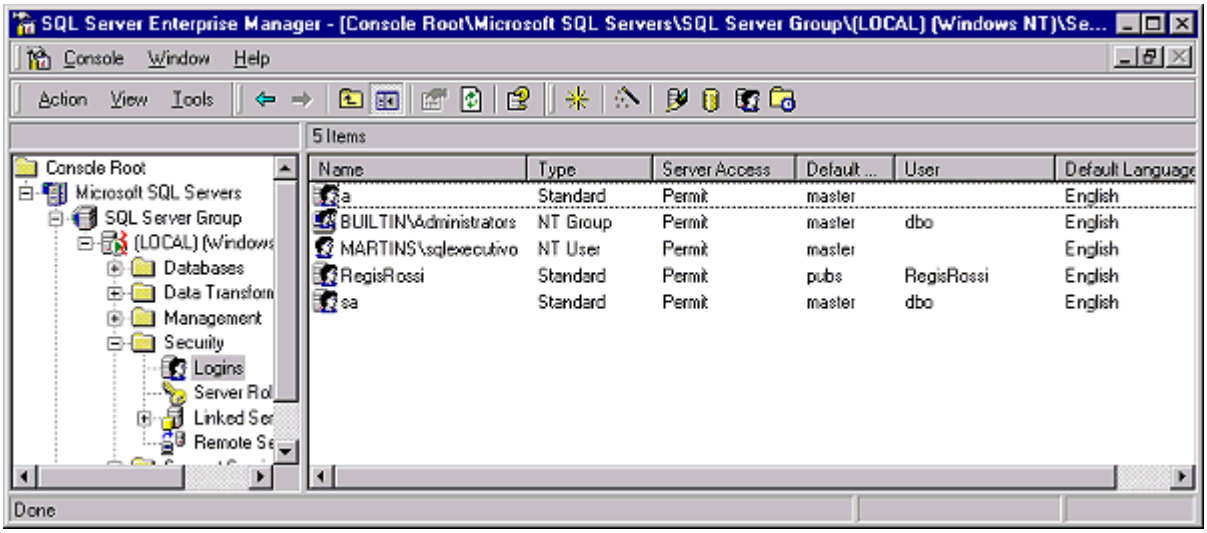
Você pode adicionar logins aos papéis de servidor sempre que achar necessário. Mas lembre-se que o ato de delegar privilégios administrativos a usuários às vezes pode ser

arriscado, e você não irá querer dar privilégios demais para usuários. Apenas dê aos logins os privilégios absolutamente mínimos que eles precisam para completar as tarefas que você os atribuiu.

Visualizando informações de segurança

Visualizando informações de logins do SQL Server

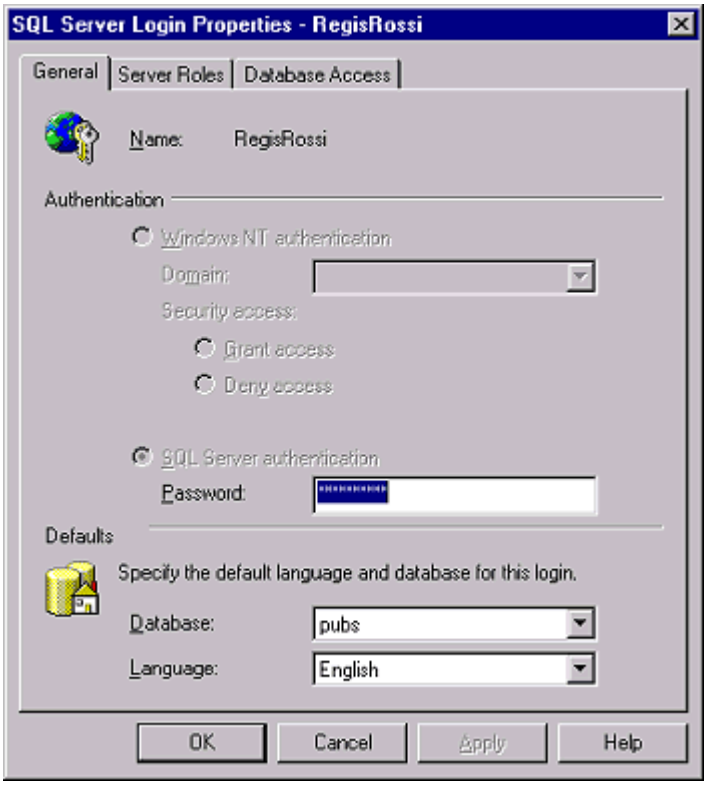
No Enterprise Manager, expanda o servidor cujas contas você quer obter informações, clique na pasta Security, e então em logins. Aparece uma tela semelhante à mostrada abaixo:



Observe na parte direita da tela estas informações:

- A coluna "Name" mostra cada login existente. Se algum login tiver um nome de domínio antes do nome do login, como acima em "MARTINS\Sqlexecutivo", significa que a conta usa a autenticação do Windows NT. Os que não são precedidos por um nome de domínio usam a autenticação do SQL Server, como "a" e "sa" na figura acima.
- A coluna "Type" dá mais informações sobre o login. Se o tipo é "NT User", a conta foi o NT Server e adicionada como um login do SQL Server. Se o tipo é "NT Group", isso significa que qualquer usuário que faça parte desse grupo do Windows NT pode acessar o SQL Server utilizando sua conta de grupo como login. Se o tipo é "Standard", esse login foi criado usando com o Enterprise Manager.
- A coluna "Default Database" mostra qual banco de dados cada usuário usa como seu banco de dados padrão. É o banco de dados no qual eles são automaticamente logados quando eles acessam o SQL Server pela primeira vez.
- A coluna "User" mostra o nome de usuário que este usuário recebeu no banco de dados padrão (o que ele é automaticamente logado da primeira vez que efetua logon no SQL Server.
- A coluna "Default Language" mostra a língua específica para o login. O padrão é inglês [English].

Para ver informações específicas sobre qualquer um dos logins, clique no mesmo com o botão direito, e selecione Properties. Aparece a tela abaixo:



Aqui você pode ver e configurar quase todas opções de login. Essa tela tem três guias.

Na guia General, você pode alterar a senha para esse login [Password], e definir seu banco de dados e linguagem padrão ([Database] e [Language]).

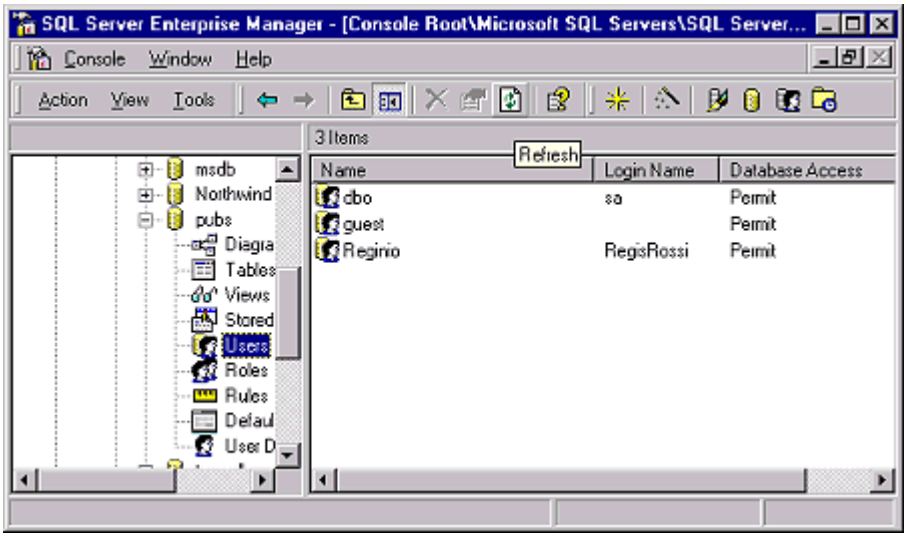
Na guia "Server Roles" mostra a quais papéis de servidor o login pertence. A guia "Database Access" mostra a quais bancos de dados o login tem acesso (ou seja, tem um login definido na tabela *syslogins* do banco de dados), além de mostrar a quais papéis de banco de dados o usuário pertence, em cada banco de dados.

Clique em Cancel para sair da jabela de propriedades do login.

Visualizando informações de IDs de usuário do banco de dados

Além de ver as informações de cada um dos logins definidos para o SQL Server, também é possível ver as informações dos IDs de usuário definidos para cada banco de dados.

Para isso, no Enterprise Manager, selecione o banco de dados cujas informações de IDs de usuário você quer ver, expanda-o e clique em **Users**.



Note que no lado direito da tela aparecem algumas informações sobre os IDs definidos para o banco de dados:

- A coluna Name mostra o ID de usuário que foi adicionado a este banco de dados, indicando quem tem a capacidade de acessar este banco de dados.
- A coluna "Login Name" mostra qual login está associado com os IDs de usuário definidos para esse banco de dados.
- A coluna "Database Acces" indica o tipo de acesso que o ID de usuário tem a esse banco de dados.

Selecione, do lado direito da tela, o login cujas informações você deseja ver, clique no mesmo com o botão direito e selecione Properties.

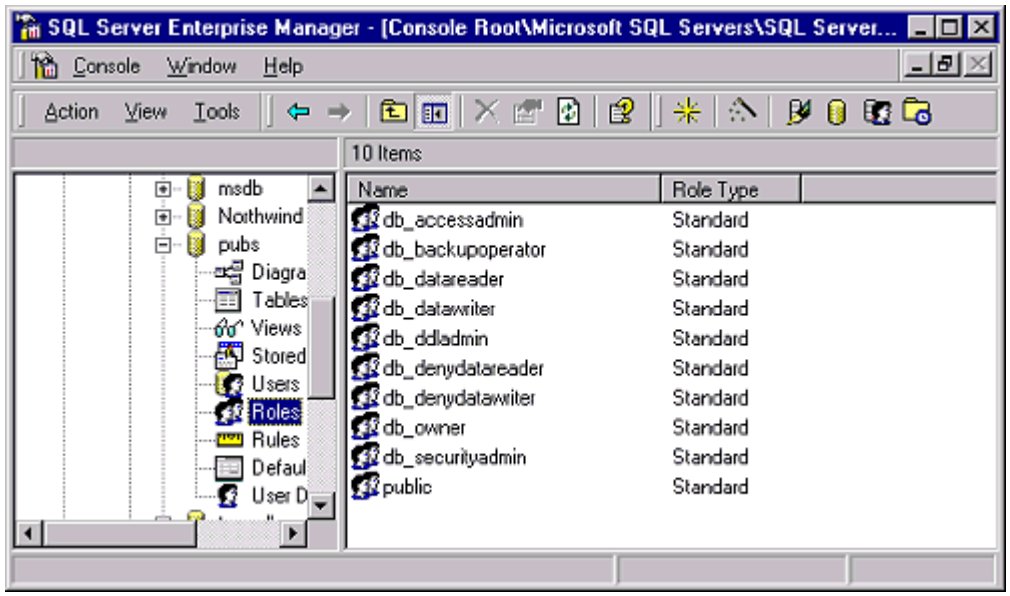
Essa tela mostra todos os papéis de banco de dados definidos para este banco de dados (todos que aparecem listados) e também a quais deles este usuário específico pertence (os que têm a caixa de verificação ao seu lado marcada).

Para sair desta janela, clique em Cancel.

Visualizando informações de papéis de bancos de dados.

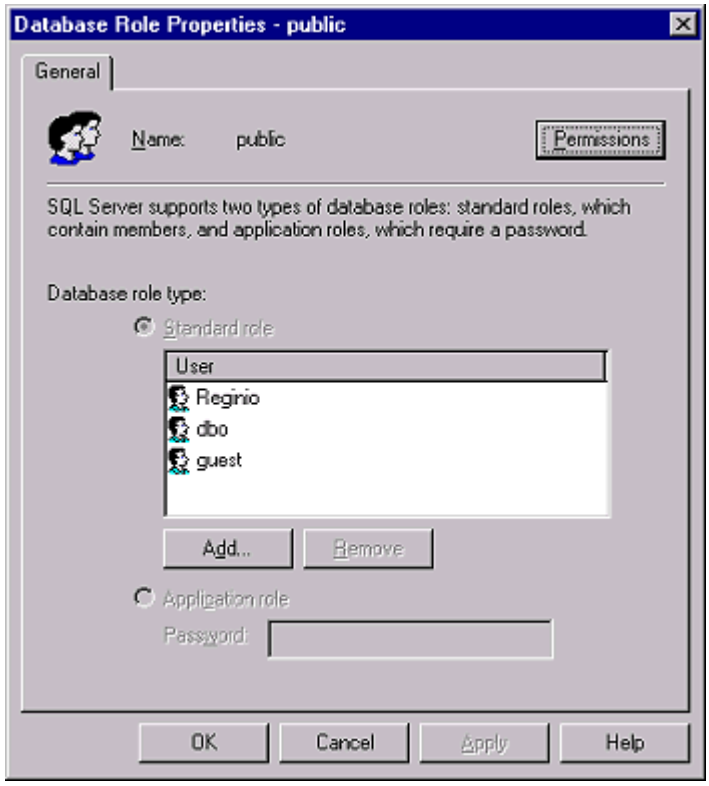
Embora você possa ver informações sobre papéis de bancos de dados com a técnica descrita acima, também pode ver-se através da perspectiva dos papéis de bancos de dados, ao invés do usuário de banco de dados.

Para isso, no Enterprise Manager, expanda o banco de dados de cujos papéis você quer obter informações, clique em **Roles**.



Na coluna "Name"você vê uma lista de todos os papéis para esse banco de dados particular. Na coluna "Role Type" vê-se as palavras "Standard" ou "Application". Standard significa que é um papel normal de banco de dados, ebquanto Application significa que esse papel é um papel de aplicação de banco de dados.

Caso você queira obter mais informações sobre qualquer dos papéis, clique no mesmo com o botão direito e selecione **Properties**.



Essa caixa de diálogo lista os usuários do banco de dados que fazem parte deste papel em particular.

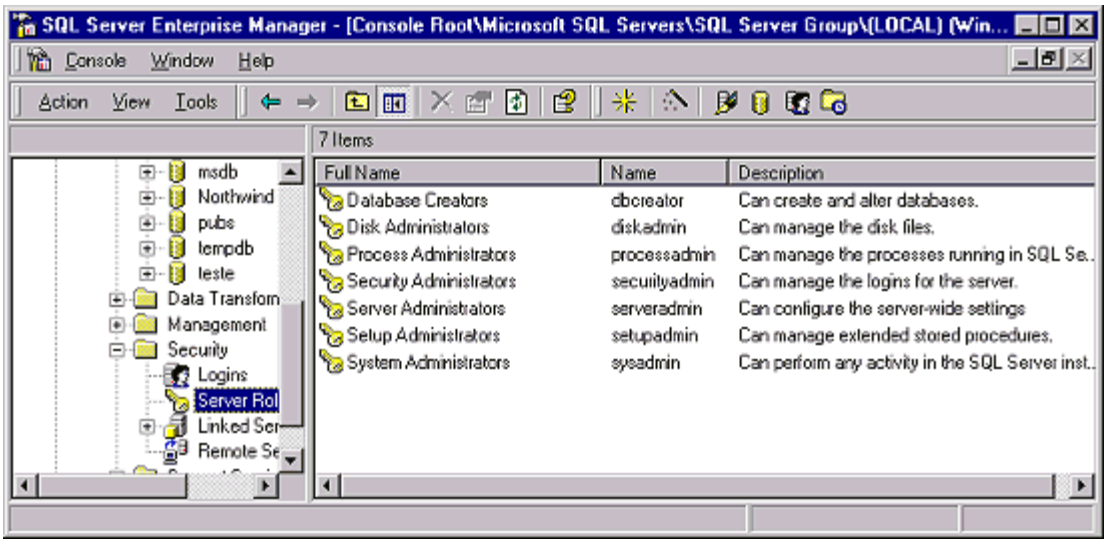
Para sair dessa caixa de diálogo, clique em Cancel.

É bem provável que você ache mais fácil ver estas informações através das informações de login, como descrito anteriormente.

Visualizando informações de papéis de servidor

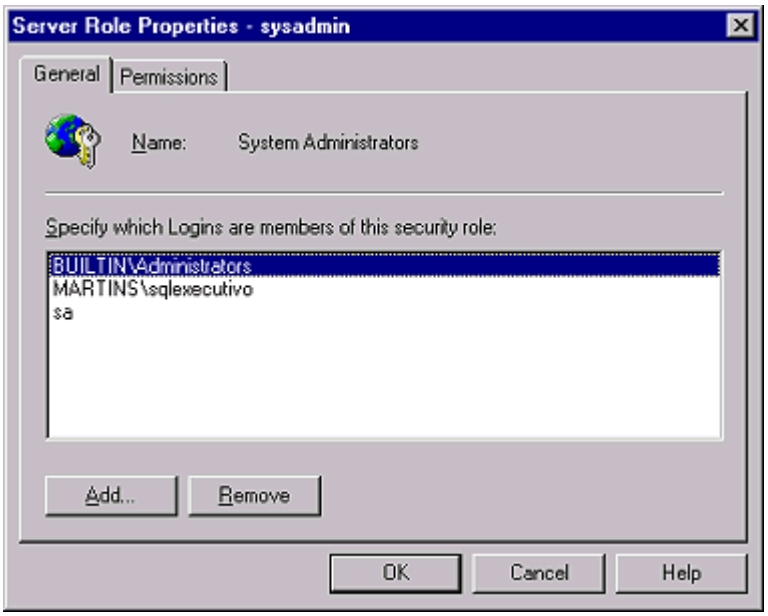
Muitas vezes, você irá querer ver os vários papéis de servidor de seu SQL Server e determinar quais logins pertencem a quais papéis de servidor.

No Enterprise Manager, selecione o servidor cujos papéis você quer gerenciar, e expanda-o. Expanda a pasta **Security** e selecione **Server Roles**.



O nome completo do papel de servidor é mostrado na coluna "Full Name", e o seu nome curto em "Name". A coluna "Description" descreve o que o papel de servidor pode fazer.

Para descobrir quais logins pertencem a cada um dos papéis de servidor, clique com o botão direito no papel de servidor, e selecione Porperties. Aparece a caixa de diálogo de "Propriedades do papel de servidor".



Essa caixa de diálogo tem duas guias: a guia "General", que te diz quais logins foram atribuídos a esse papel particular. A guia "Permissions" te mostra as várias permissões que esse papel recebeu.

Clique em Cancel para sair dessa janela.

Nota: Esse é o único local onde se pode ver informações sobre papéis de servidor.

Permissões

Até agora, já vimos como criar e gerenciar logins que são usados para controlar o acesso ao SQL Server. Vimos também como criar e gerenciar IDs de usuários de bancos de dados, os quais são usados para controlar o acesso a bancos de dados individualmente. Mas, mesmo que um usuário tenha um login e um ID de usuário válido, ele não pode acessar qualquer dado em um banco de dados sem que lhe tenham sido dadas permissões explícitas para acessar os objetos armazenados no banco de dados.

Permissões são usadas no SQL Server para especificar quais usuários podem ter acesso a quais objetos de bancos de dados, e o que eles podem fazer com tais objetos. Se um usuário não receber explicitamente a permissão para acessar um objeto, ele não terá acesso ao mesmo. Permissões podem ser atribuídas a *usuários* (contas do NT Server ou do SQL Server), *grupos* (grupos globais do NT Server), e *papéis* (papéis predefinidos de servidor, de banco de dados e papéis personalizados de bancos de dados). É mais fácil atribuir permissões a grupos e papéis do que a usuários individuais (a quantidade de trabalho braçal exigida é menor).

O SQL Server apresenta três níveis de permissões:

- *Permissões para comandos SQL*: habilitam usuários a executar comandos SQL específicos que são usados para criar objetos de bancos de dados, fazer backup de bancos de dados e logs de transação.
- *Permissões de objetos*: determinam o que um usuário pode fazer a um objeto preexistente.
- *Permissões implícitas*: são permissões que só podem ser executadas por membros de papéis predefinidos de servidor e de banco de dados, ou pelos proprietários do banco de dados.

Atribuem-se permissões aos usuários baseado no que eles precisam de fazer com os dados armazenados no SQL Server. Alguns usuários podem precisar apenas de visualizar dados, outros podem precisar de consultar dados e gerar relatórios, outros podem precisar de alterar dados, etc. Uma das principais responsabilidades do DBA é determinar quais usuários precisam de acessar quais objetos, e quais permissões eles precisam.

Permissões atribuídas em um banco de dados são independentes de permissões atribuídas a outro banco de dados. Se um usuário precisar de acessar tabelas em dois bancos de dados, o usuário deve ter IDs de usuário nos dois bancos de dados, e as permissões necessárias atribuídas em cada banco de dados, para acesso aos objetos que ele precisa acessar.

Permissões para comandos SQL

Permissões para comandos SQL são dadas a usuários que precisam de criar um banco de dados ou objetos de bancos de dados, ou que precisam fazer backup de bancos de dados e seus logs de transações. Quando você atribui permissões para comandos SQL você na verdade está dando àquele usuário específico a capacidade de executar comandos SQL específicos. Esses comandos são os seguintes:

- CREATE DATABASE: capacita o usuário a criar bancos de dados em um servidor SQL Server específico.
- CREATE DEFAULT: o usuário pode criar um valor padrão que é automaticamente inserido em uma coluna de alguma tabela sempre que a coluna for deixada em branco quando um novo valor é acrescentado a ela.
- CREATE PROCEDURE: permite a criação de procedimentos armazenados.
- CREATE RULE: permite a criação de uma regra que é utilizada para validar dados que são informados em uma coluna sempre que uma nova linha é adicionada à tabela.
- CREATE TABLE: permite a criação de uma nova tabela dentro de um banco de dados
- CREATE VIEW: permite a criação de tabelas virtuais, que são usadas para mostrar um subconjunto de uma tabela, ou para juntar duas ou mais tabelas em uma única tabela virtual.
- DUMP DATABASE: permite fazer backup de um banco de dados.
- DUMP TRANSACTION: permite fazer backup do log de transações de um banco de dados.

As tarefas descritas acima podem ser realizadas diretamente através de comandos SQL, ou usando o Enterprise Manager. Você pode atribuir a um usuário uma única permissão por vez, todas elas, ou um conjunto das permissões de comando disponíveis.

Na realidade, raramente serão usadas as permissões para comandos SQL, pois o SQL Server já inclui papéis que cumprem as mesmas funções que a atribuição dessas permissões. Por exemplo, o papel predefinido de servidor Sysadmin consegue reaIizar qualquer tarefa que possa ter sido atribuída a um usuário através de permissões para comandos SQL. Assim como o papel predefinido de banco de dados db_backupoperator pode fazer os backups de um banco de dados da mesma maneira que quem recebeu a permissão DUMP DATABASE. O mais prático é atribuir os usuários a papéis de servidor ou de banco de dados que lhe permitam fazer as tarefas que forem necessárias.

Permissões de objetos

O tipo mais comum de permissão atribuído a usuários, grupos e papéis é a permissão de objetos. Essas permissões determinam quem pode acessar um objeto preexistente e o que esse usuário pode fazer com tal objeto. Quando você atribui a um usuário uma permissão de objeto, você na verdade está dando a tal usuário a capacidade de executar certos comandos SQL sobre objetos em um banco de dados. Essas permissões são as seguintes:

- DELETE: permite excluir uma tabela ou visão em um banco de dados.
- EXECUTE: premite a execução de um procedimento armazenado.
- INSERT: permite adicionar-se uma nova linha em uma tabela, ou em uma tabela através de uma visão.
- REFERENCES: (DRI) permite ligar duas tabelas usando uma coluna comum.
- SELECT: permite pesquisar e visualizar dados de uma visão, tabela ou coluna.
- UPDATE: permite modificar dados em uma tabela, coluna de uma tabela, ou em uma tabela através de uma visão.

As tarefas relacionadas a objetos citadas acima podem ser executadas com o Enterprise Manager, ou pelo uso de comandos SQL, ou indiretamente através do uso de qualquer aplicação "front-end" de cliente que use comandos SQL para acessar dados do SQL Server em um servidor. Independente de como um usuário acessa objetos em um banco de dados, cada usuário deve receber explicitamente permissões, em cada objeto, para realizar o acesso.

Permissões implícitas

Uma permissão implícita é uma permissão que um usuário obtém apenas pelo fato de pertencer a um papel predefinido de banco de dados ou de servidor, ou por ser o proprietário de um objeto de banco de dados. Permissões implícitas não podem ser atribuídas a usuários. Ao invés disso, um usuário que precise de uma permissão implícita deve ser adicionado a um papel predefinido que já tenha tal permissão. Permissões implícitas podem assim ser atribuídas a usuários, papéis personalizados ou grupos, com a simples atribuição dos mesmos a um papel predefinido de banco de dados ou de servidor.

As permissões implícitas também podem ser atribuídas a usuários, grupos ou papéis personalizados definindo quaisquer destes como o proprietário de um objeto de banco de dados específico.

Os usuários, grupos ou papéis personalizados podem ser atribuídos a qualquer um dos **Papéis predefinidos de Servidor** ou **Papéis predefinidos de Banco de Dados**, recebendo as permissões que tal papel tenha. (relembre quais são os Papéis predefinidos de Servidor e Papéis predefinidos de Banco de Dados)

Além de receberem permissões através da atribuição aos papéis acima, também podemos fazer usuários tornarem-se proprietários de algum objeto. Como funciona isso? Quando um usuário com a permissão de comando adequada cria um novo objeto no banco de dados, tal como uma tabela, ele se torna o *proprietário do objeto de banco de dados* [Database object owner] (DBOO) daquele objeto. Proprietários de objetos de banco de addos têm permissões implícitas em todos os objetos que lhes pertençam, o que os dá a capacidade de executar qualquer atividade naquele objeto, tal como SELECT, INSERT, UPDATE, DELETE, entre outros. Eles têm controle completo dos objetos que criam. Como dá para perceber, permitir que qualquer um seja um DBOO não é uma boa idéia. Normalmente, as únicas pessoas que devem criar objetos de bancos de dados são DBAs ou desenvolvedores SQL, não usuários comuns.

Precedência de permissões

Cinco níveis de permissões podem ser atribuídas a um usuário, conforme segue:

- Permissões individuais
- Permissões de grupos globais do NT Server
- Permissões de papéis predefinidos de servidor
- Permissões de papéis predefinidos de bancos de dados
- Permissões de papéis personalizados de bancos de dados.

As permissões podem ser dos tipos: implícita, de comandos ou de objetos.

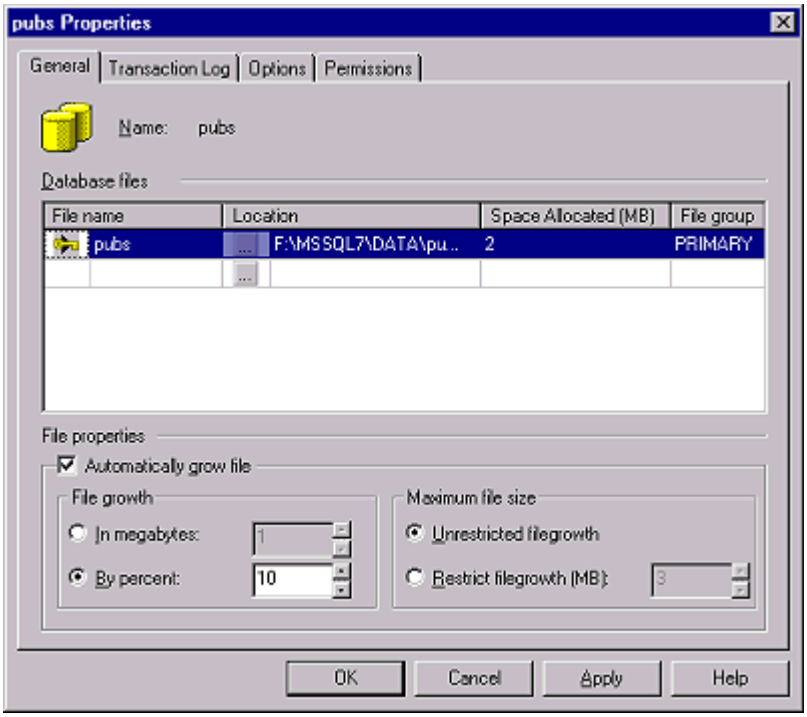
O que ocorre se um usuário receber permissões diferentes através de várias permissões individuais, vários grupos ou papéis de que o mesmo faça parte?

A priori, as permissões somam-se, ou seja: as permissões que um usuário tenha como membro de um grupo somam-se às permissões que ele tiver como usuário individual e assim por diante. Mas há uma exceção! A permissão "negar acesso" [deny access] sobrepõe-se a qualquer outra permissão para o objeto em questão. Quer dizer que se um usuário tiver obtido permissão para visualizar dados de uma tabela, através da permissão de comando SELECT para a tabela, e o mesmo usuário fizer parte de um grupo global que tem a permissão de "acesso negado" à tabela em questão, sua permissão efetiva será a de "acesso negado", ou seja, não lhe será permitido acessar tal banco de dados.

Apesar de termos exemplificado aqui citando uma tabela, essa regra é válida para qualquer objeto de banco de dados.

Visualizando informações de permissões

Antes que você aprenda a conceder e revogar permissões para usuários, grupos ou papéis, é importante que você saiba como visualizar permissões tanto de objetos como de

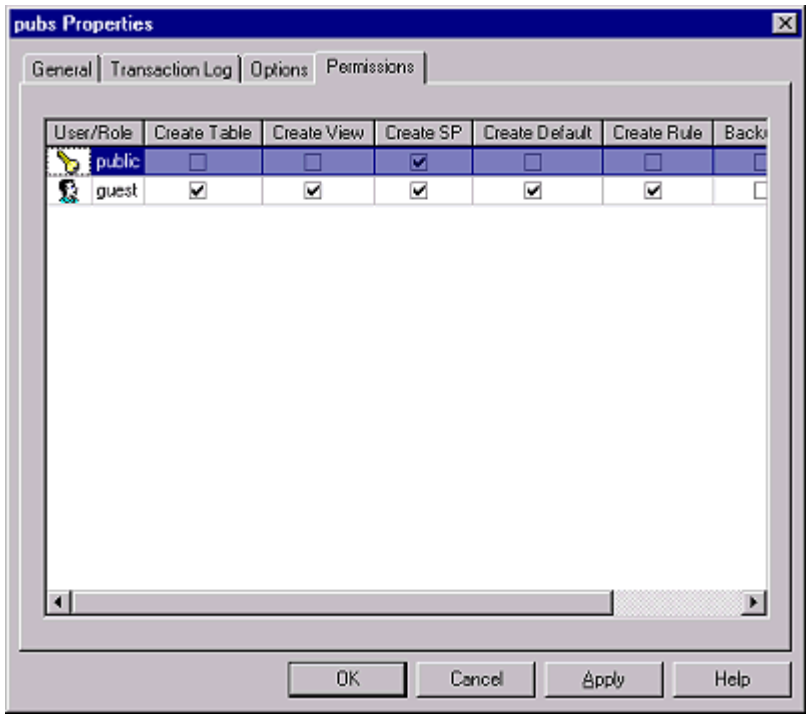


comandos. Isso não apenas te ajudará a trabalhar com permissões, mas também te mostrará como executar uma tarefa que você estará realizando regularmente como um DBA. Vamos ver como visualizar as permissões atuais de objeto para todos os usuários, grupos, e papéis em um único banco de dados utilizando o Enterprise Manager. Lembre-se que permissões são gerenciadas para cada banco de dados, e que você deve realizar estes passos em cad banco de dados no qual você queira ver as permissões.

Visualizando permissões para comandos SQL

No Enterprise Manager, usando uma conta com privilégios de *sysadmin*, expanda o banco de dados cujas permissões de comando você quer visualizar. Clique no mesmo com o botão direito e em **Properties**. Aparece a caixa de diálogo de Propriedades do banco de dados:

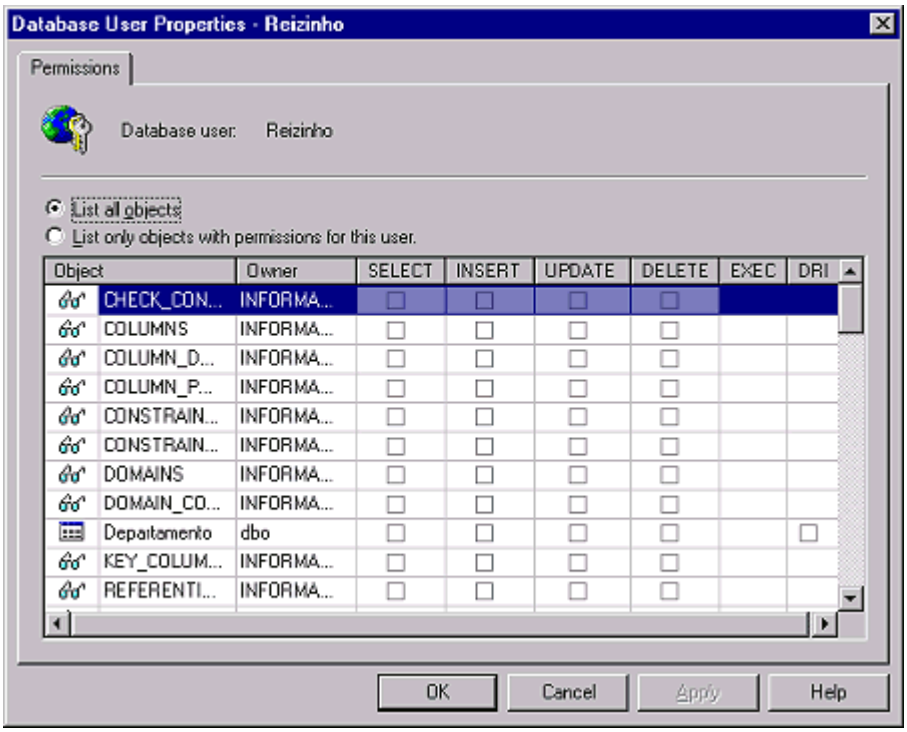
Agora, clique na guia **Permissions**. É mostrada a tela de permissões para comandos SQL



Na primeira coluna desta tela, embaixo do título **User/Role**, estão listados toos os IDs de usuários de bancos de dados para esse banco de dados. Lembre-se que essa coluna pode exibir quaisquer grupos, papéis ou usuários. Nas outras colunas estão as várias permissões para comandos SQL que podem ser atribuídas. Note que esta tela não exibe todas as permissões de uma vez; você deve percorrê-la para a direita para poder vê-las todas. Depois de ver todas as permissões que podem ser atribuídas, saia desta tela clicando em **Cancel**. Isso te leva de volta à tela do Enterprise Manager.

Visualizando permissões de objetos

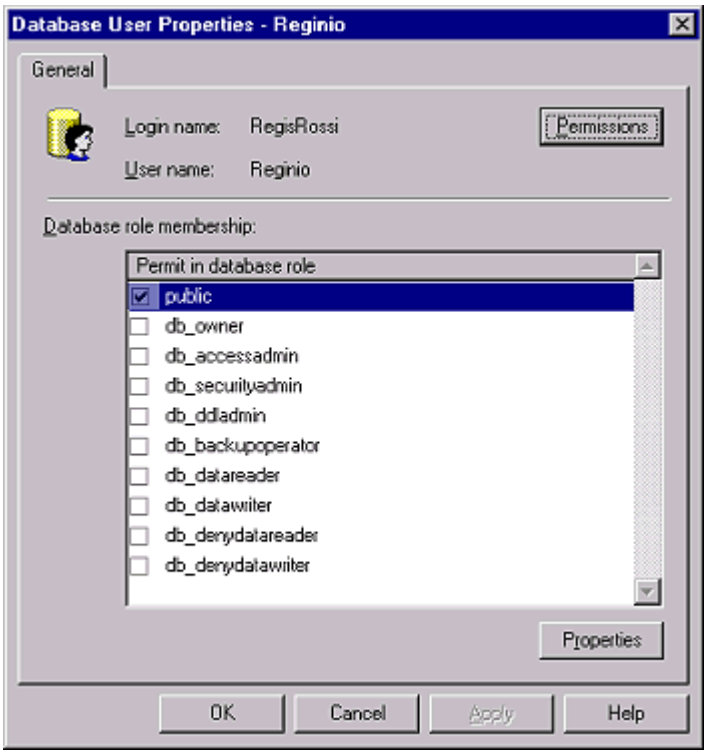
A visualização de permissões de objetos é m pouco mais difícil do que a visualização de permissões para comandos SQL. Você pode vê-las da perspectiva de usuários, grupos, ou papéis, ou então da perspectiva dos próprios objetos. Analisaremos aqui as duas maneiras.



Sob a perspectiva do usuário, grupo ou papel, as permissões são visualizadas desta forma:

12. No Enterprise Manager, expanda o banco de dados cujas permissões de objetos você quer visualizar.
13. O próximo passo depende se você quer ver permissões de objetos para grupos e usuários, ou para papéis personalizados.
    - Caso você queira ver as permissões de objetos para grupos e usuários, selecione **Users** no banco de dados em questão; todos os usuários e grupos aparecem do lado direito da tela.
    - Para ver informações de permissões de objetos atribuídas a papéis personalizados, selecione **Roles** no banco de dados em questão; todos os papéis, predefinidos e personalizados são mostrados no lado direito da tela.

14. Agora, no lado direito da tela, clique com o botão direito em um usuário, grupo ou papel personalizado, cujas permissões de objetos você quer visualizar,e então selecione Properties. Aparece então a janela de propriedades do usuário ou do papel (dependendo do que você selecionou no passo 2).



15. Clique no botão Permissions para ver as permissões a nível de objeto que esse usuário (ou papel) tem.

Veja que há dois botões no topo da tela. Quando o primeiro [List all objects] está selecionado, todos os objetos pertencentes ao banco de dados são exibidos na tela. Se a segunda opção [List only objects with permissions for this user] for selecionada, apenas aqueles objetos para os quais o usuário tm permissão são listados.

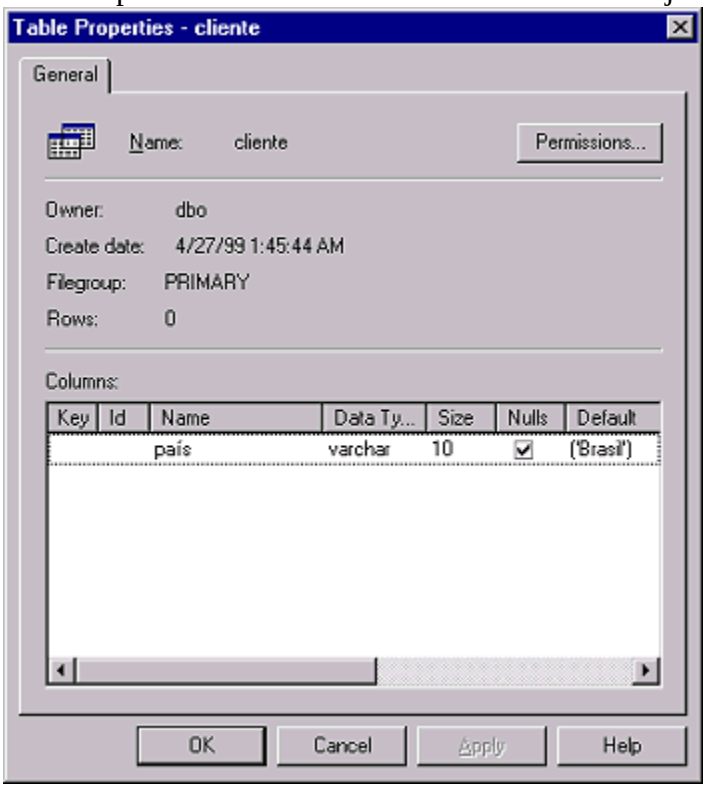
Na primeira coluna há um ícone que representa o objeto de banco de dados. A segunda coluna, lhe mostra o nome do objeto. A coluna Owner mostra quem é o proprietário do objeto. As outras colunas mostram as permissões de objetos disponíveis. Se alguma coluna estiver marcada (com sua caixa de verificação ativada), isso indica que esse usuário possui aquela permissão para o objeto em questão.

Perceba que nem todos os objetos têm todas as permissões de objeto disponíveis. Por exemplo, procedimentos armazenados têm apenas a permissão de objeto **Execute**.

16. Para terminar de visualizar as permissões de objeto, saia da tela clicando em **Cancel**. Clique de novo em **Cancel** e você estará de volta ao Enterprise Manager.

Sob a perspectiva dos objetos individuais de banco de dados, visualizam-se assim as permissões:

1. No Enterprise Manager, expanda o banco de dados cujas permissões de objeto você quer verificar. Aparecem todos os tipos de objetos de bancos de dados.
2. Agora você deve decidir quais permissões de objetos você quer visualizar. Você pode escolher: tabelas [tables], visões [views], procedimentos armazenados [stored procedures], regras [rules], defaults [defaults], e tipos de dados definidos pelo usuário [user defined data types]. Clique no tipo de objeto ^cujas permissões você quer visualizar. Aparecem no lado direito da tela todos os objetos desse tipo.



3. Clique com o botão direito em um dos objetos, e em **Properties**. Aparece a caixa de diálogo de propriedades do objeto que você selecionou (no caso uma tabela)

4. Para exibir as permissões para esse objeto, clique no botão Permissions. Aparece a tela de permissões do objeto, como abaixo:

Note as duas opções na parte superior da tela. Por padrão, a primeira [List all users / user-defined DB roles / public] está selecionada. Assim, todos usuários, grupos e papéis para esse banco de dados são exibidos na tela. Se a segunda opção [List only users / user-defined DB roles / public permissions on this object], apenas os usuários, grupos ou papéis que tenham permissões definidas para esse objeto serão exibidos.

A primeira coluna mostra um ícone. Uma única cabeça indica um usuário ou um grupo. Duas cabeças indicam um papel. Todos os usuários, grupos ou papéis para esse banco de dados estão embaixo de "User/DB Role". As colunas restantes indicam as permissões de objeto disponíveis para este objeto. Se a caixa de verificação estiver selecionada (ativa),

indica que um usuário, grupo ou papel obteve a permissão de objeto associada. Veja que nem todos os objetos têm todas as permissões de objeto.

5. Depois de terminar de visualizar as permissões de objeto, você pode sair clicando em **Cancel** duas vezes, primeiro para a tela de permissões, e depois para a caixa de diálogo de propriedades. Então você volta para a tela principal do Enterprise Manager.

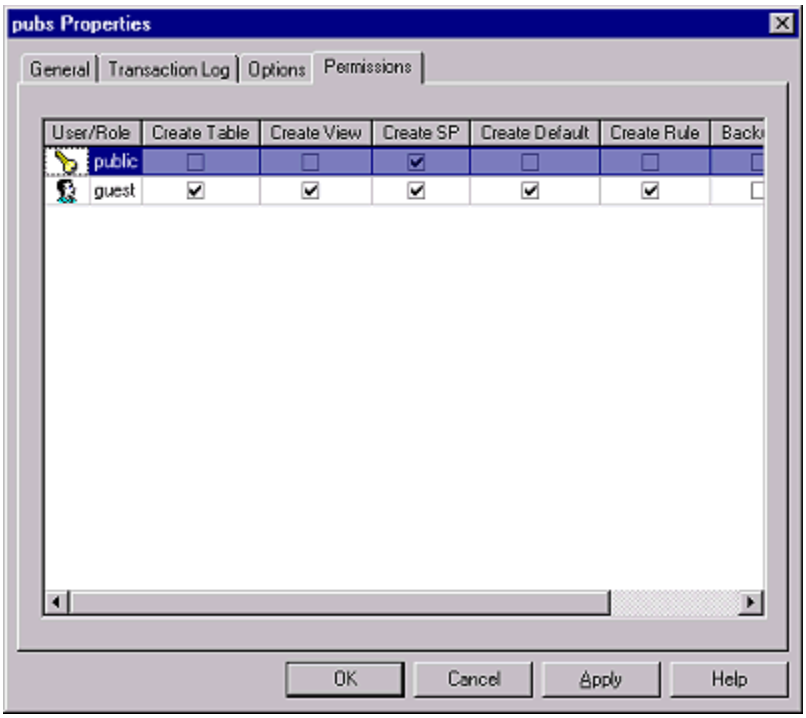
Concedendo e revogando permissões para comandos SQL pelo Enterprise Manager

A concessão e revogação de permissões usa as mesmas telas que acabamos de ver.

Mas agora, atribuíremos e revogaremos permissões de grupos, usuários, e papéis.

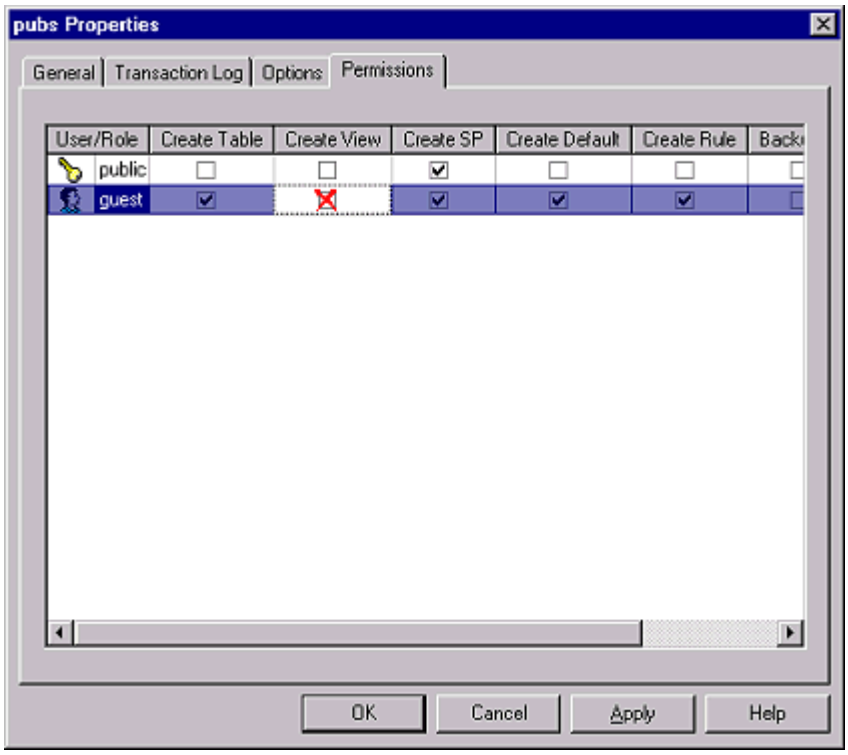
Lembre-se que nenhum usuário tem qualquer permissão para acessar qualquer objeto de dados até que você explicitamente atribua a ele tais permissões. Quando você concede uma permissão de comandos para um usuário, você está lhe dando a permissão de executar uma tarefa específica, tal como criar objetos de banco de dados ou fazer backup de um banco de dados ou de um log de transações. Esse usuário permanece com a permissão que você lhe deu até que e;a seja explicitamente removida. Depois que uma permissão for revogada, o usuário não pode mais realizar a mesma tarefa, até que lhe tenha sido concedida a mesma permissão de comando novamente.

Para conceder ou revogar uma permissão utilizando o Enterprise Manager, os passos são os seguintes:



1. No Enterprise Manager, clique com o botão direito no banco de dados cujas permissões você quer alterar, e em **Properties**. Aparece a caixa de diálogo abaixo:

2. Essa é a mesma tela vista anteriormente (em visualizando permissões de bancos de dados). Na primeira coluna desta tela, abaixo de **User/Role** estão listados todos os IDs de usuário de banco de dados para este banco de dados. Lembre-se que esta coluna pode listar qualquer usuári, grupo ou papel. Nas outras colunas estão as várias permissões para comandos SQL que podem ser atribuídas. Note que na tela não cabem todas as permissões existentes; para vê-las, você tem que rolar horizontalmente para a direita.
3. Para atribuir qualquer das sete permissões para comandos SQL para qualquer usuário, papel ou grupo exibidos na primeira coluna, clique na coluna da permissão que você quer atribuir, na linha do usuário que deve receber tal permissão. A permissão não é concedida até que você clique em Apply ou Ok.



4. Para revogar uma permissão de comando que tenha sido atribuída anteriormente, clique na caixa de verificação que representa a permissão de comando que você quer revogar do usuário, grupo, ou papel. Quando você clicar na caixa de verificação, ela muda para um X vermelho (como mostrado abaixo), indicando que a permissão será revogada. A permissão só é de fato revogada quando você clica em Ok ou Apply.

5. Depois de revogar e conceder todas as permissões para comandos SQL que você queira, saia dessa tela clicando em Ok. Então você volta para o Enterprise Manager.

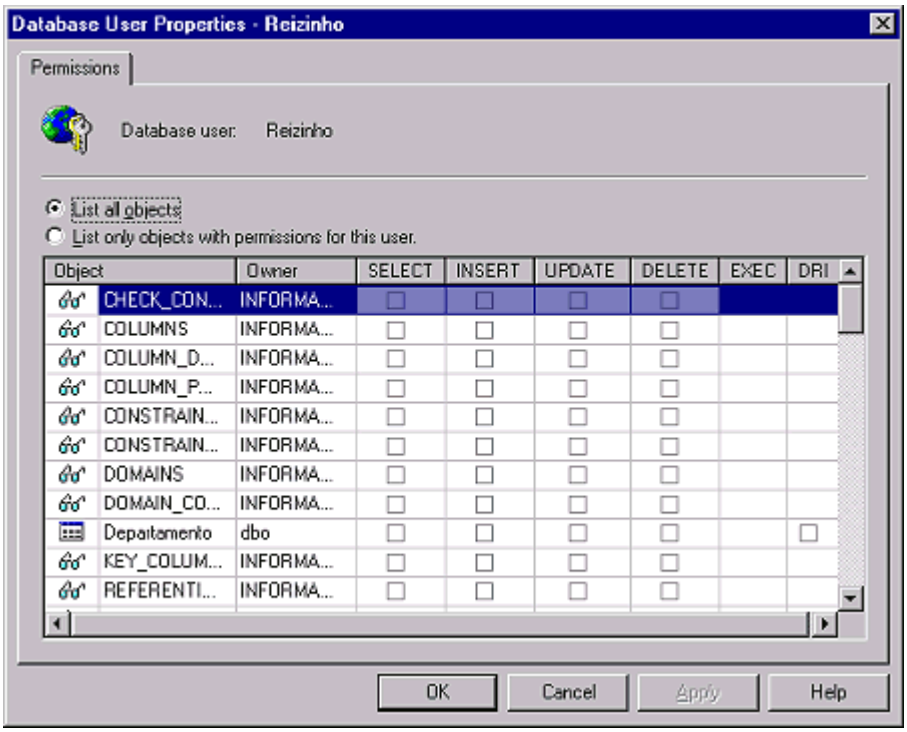
Concedendo e revogando permissões de objetos pelo Enterprise Manager

Quando mostramos como visualizar permissões de objetos para usuários, grupos ou papéis, vimos que há dois modos de visualizá-las. Pode-se ver as permissões de objeto sob a perspectiva do usuário, grupo ou papel, ou a pela perspectiva do objeto de banco de dados em si. Isso também se verifica para a concessão e revogação de permissões de objeto. Aqui, demonstraremos como conceder/revogar permissões de objetos pela perspectiva do usuário, grupo ou papel, pois essa é a maneira mais prática e conveniente.

Não se esqueça de que um usuário não tem permissão para acessar qualquer objeto de banco de dados até que tal permissão lhe tenha sido atribuída Quando você concede uma permissão de objeto a um usuário, você está lhe dando a permissão de executar uma tarefa em um objeto preexistente, tal como SELECT, INSERT, UPDATE, ou DELETE sobre o objeto. Esse usuário mantém a permissão de objeto até que a mesma tenha sido explicitamente revogada.

Caso você queira remover/conceder permissões pela perspectiva do objeto de banco de dados, você pode, usando praticamente os mesmos procedimentos que serão descritos a seguir.

1. Expanda o banco de dados cujas permissões de objeto você quer alterar. Se você for conceder/revogar permissões para usuários ou grupos, clique em **Users**. Caso você queira alterar permissões para papéis, clique em **Roles**.
2. Clique com o botão direito no usuário, grupo ou papel cujas permissões você quer alterar, e em **Properties**. Aparecerá uma caixa de diálogo com propriedades do



usuário ou grupo, ou do papel.

3. Clique em **Permissions**. Aparece a tela de permissões para o usuário, grupo ou papel que você houver selecionado.

   A primeira coluna tem um ícone que representa o tipo do objeto de banco de dados, seguido do nome do objeto de banco de dados. A coluna Owner mostra quem é o proprietário do objeto. As outras colunas mostram as permissões de objeto existentes. Uma marcação em alguma das caixas de verificação indica que o usuário em questão tem permissão para esse objeto.
4. Para conceder alguma das seis permissões de objetos para o usuário, grupo ou papel em questão, marque a caixa de verificação apropriada na coluna referente ao objeto. A caixa de verificação ficará marcada. A permissão só é de fato concedida quando você clica em Ok ou Apply.
5. Para revogar uma permissão que já tenha sido atribuída, clique na caixa de verificação que representa a permissão de objeto que você quer remover de um usuário, grupo ou papel. Ao clicar na mesma, ela muda para um X vermelho, indicando que a permissão será revogada. A permissão só é de fato revogada ao se clicar em Ok ou Apply.
6. Depois de teminar de definir as permissões de objetos, você pode sair da tela clicando em Ok. Você deve então clicar em Ok de novo para retornar ao Enterprise Manager.

Concedendo e revogando permissões para comandos SQL, usando comandos SQL

Permissões também podem ser concedidas ou revogadas através de comandos SQL.

Para isso, usa-se os comandos GRANT e REVOKE.

GRANT concede permissões, enquanto REVOKE as revoga. A sintaxe do GRANT é:

```
GRANT {ALL | comando [,..n]}
TO conta_segurança [,..n]
```

E a do REVOKE é:

```
REVOKE {ALL | comando[,...n]}
FROM conta_segurança [,...n]
```

Onde:
comando é o comando SQL para o qual a permissão está sendo concedida/removida. Os comandos podem ser:

- CREATE DATABASE
- CREATE DEFAULT
- CREATE PROCEDURE
- CREATE RULE
- CREATE TABLE
- CREATE VIEW
- BACKUP DATABASE

- BACKUP LOG

ALL indica que todas as permissões da(s) conta(s) de segurança em questão serão concedidas/revogadas.
conta_segurança é a conta de segurança no banco de dados atual para a qual as permissões estão sendo adicionadas ou removidas. Pode ser um:

- Usuário do SQL Server ou do NT
- Grupo do NT
- Papel do SQL Server

n indica que pode ser informado mais de um nome de conta de segurança para se conceder/revogar permissões, assim como pode-se informar mais de um comando para tr permissão concedida/revogada. Basta separá-los por vírgula.

Caso quiséssemos por exemplo, permitir que um usuário pudesse criar uma tabela, digitaríamos

```
GRANT create table TO usuario
```

E para revogar essa permissão:

```
REVOKE create table FROM usuario
```

Para revogar todas as permissões de um usuário, digitaríamos este comando:

```
REVOKE ALL FROM usuario
```