Apostila Microsoft SQL Server 7.0

1 - Introdução

Visão geral do SQL Server

Sistemas Gerenciadores de Bancos de Dados

Objetivos:
- Obter uma visão geral do SQL Server e do seu funcionamento;
- Conhecer as ferramentas do SQL Server;
- Saber a divisão de papéis entre o administrador do sistema e o implementador de bancos de dados.

Visão Geral do SQL Server

O SQL Server é um sistema de gerenciamento de bancos de dados cliente/servidor de alto desempenho com alta integração com o Windows NT. Suas características são:
Integração com os serviços de *multithreading* [múltiplas linhas], agendamento, Monitor de Desempenho, e log de eventos do Windows NT. Um usuário pode se conectar ao SQL Server com a mesma senha usada para a rede Windows NT.
Replicação nativa permite disseminar informações para vários locais, reduzindo a dependência de um servidor único, e deixando a informação necessária mais próxima de quem realmente precisa dela.
Arquitetura paralela, que executa as funções de banco de dados simultaneamente para diversos usuários e tira proveito de sistemas com múltiplos processadores.
Gerenciamento centralizado de todos os servidores através de uma arquitetura de gerenciamento distribuída, com uma interface visual de gerenciamento.

Distributed Management Framework (DMF)

O SQL Server possui uma arquitetura distribuída de gerenciamento [distributed management framework], composta de objetos, serviços e componentes. Através dela, vários servidores podem ser gerenciados completamente a partir de qualquer local na rede. Entre outros componentes, essa arquitetura é composta de:

• *SQL-DMO*: biblioteca de objetos ActiveX que expõe interfaces para todas as funções de gerenciamento do SQL Server e pode ser usada em qualquer linguagem compatível com automação ActiveX. Permite gerenciar servidores, bancos de dados, tabelas e outros objetos relacionados ao banco de dados.

• *SQL Enterprise Manager*: ferramenta gráfica de administração que, utilizando os objetos SQL-DMO, simplifica o gerenciamento de um ambiente de múltiplos servidores.

• *Serviços SQLServerAgent e MSSQLServer:* executando no servidor NT, o serviço SQLServerAgent permite agendar tarefas, como backups, por exemplo, e definir alertas para informar quando ocorrem condições de erro diversas. O serviço MSSQLServer é o componente central, que permite inserir, atualizar e consultar dados armazenados no SQL Server.

Ferramentas de Administração

O SQL Server vem com várias ferramentas de administração que podem ser executadas a partir de um servidor Windows NT, de uma estação Windows NT Workstation, ou até mesmo a partir do Windows 95/98. São elas:

• *Enterprise Manager*: como já foi dito, gerencia vários servidores, permitindo executar qualquer tarefa relacionada ao SQL Server. Como será visto adiante, ele roda dentro MMC (Microsoft Management Console). Para executá-lo através de Iniciar, Executar, entre com a seguinte instrução:
mmc /s "pasta-base_do_SQLServer\BINN\SQL Server Enterprise Manager.MSC", substituindo pasta-base_do_SQLServer pela pasta onde você instalou o SQL Server 7. Por padrão, é C:\MSSQL7.

• *Service Manager(SQLMANGR.EXE)*: permite iniciar, pausar, continuar e parar ("finalizar") os serviços do SQL Server.

• *Query Analyzer(ISQLW.EXE):* permite administrar diretamente o SQL Server usando comandos Transact-SQL. Os comandos SQL podem ser executados interativamente, ou podem ser executados de procedimentos armazenados ou scripts.

• *Profiler (SQLTRACE.EXE):* permite monitorar toda a atividade do servidor e registrar essa atividade em arquivos de log, incluindo comandos SQL executados pelo servidor.

• *ClientNetwork Utility CLICONFG.EXE):* configura o software de acesso cliente numa estação.

• *Performance Monitor (SQLCTRS.PMC:* integra o Performance Monitor ("Desempenho do Sistema") do Windows NT com o SQL Server, para monitorar o desempenho do sistema.

• *Server Network Utility (SRVNETCN.EXE)*: permite adicionar, remover ou configurar as Net-libraries, que são os protocolos aceitos para comunicação do cliente com o servidor.

• *SQL Server Books Online:* toda a documentação do SQL Server, para consultar on-line. Permite fazer pesquisas de texto na documentação. Para executá-lo, em Iniciar, Executar, entre com HH pasta_base_do_SQLServer\BOOKS\SQLBOL.CHM, onde pasta_base_do_SQLServer é o diretório onde o SQL Server foi instalado. Por padrão, é C:\MSSQL7.

• *Uninstall SQL Server 7.0:* permite que você remova a instalação existente do SQL Server 7.0

Sistemas Gerenciadores de Banco de Dados

Um sistema gerenciador de banco de dados (SGBD) como o SQL Server é responsável por armazenar dados de forma confiável e permitir fácil recuperação e atualização desses dados. Um SGBD *relacional* armazena dados de forma relacional, isto é na forma de linhas e colunas.

Conceitos Relacionais

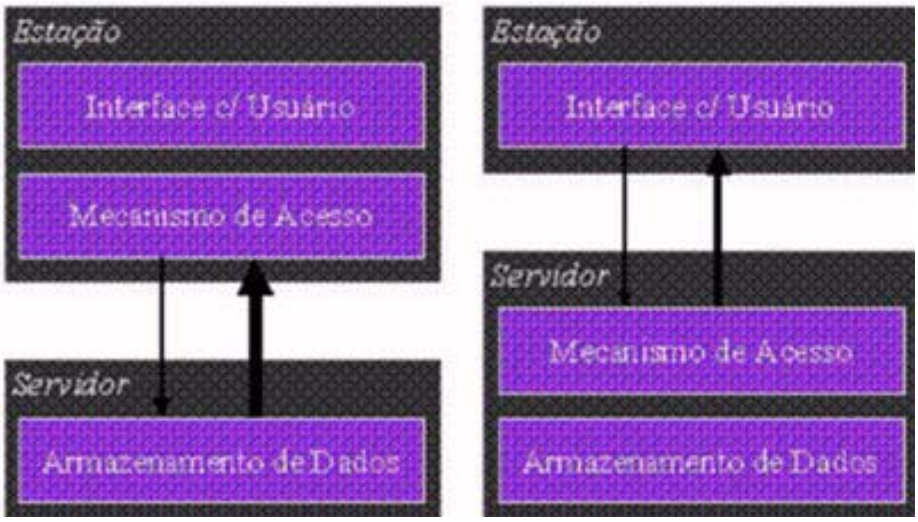
Um *registro* [record] ou *linha* [row] é um grupo de variáveis com tipos de dados diferentes, que armazenam dados relacionados. Por exemplo, um registro pode conter os dados relativos a um produto vendido pela empresa, como descrição, código de identificação, quantidade em estoque.
Um *campo* [field] ou *coluna* [column] é um dos itens de informação dentro de uma linha da tabela, como a descrição da informação.
Uma *tabela* [table] é um conjunto de linhas (registros) com a mesma estrutura, armazenados de forma permanente em disco.As tabelas são compostas de linhas(row) ou registros (record) e colunas(column) ou field(campo).
Um *banco de dados* [database] é um conjunto de tabelas que contêm dados relacionados. Por exemplo, um sistema de contas a pagar poderia ter um banco de dados de contas a pagar, com uma tabela para duplicatas, uma tabela para bancos, uma tabela para contas etc.
Um *índice* [index, plural 'indexes' ou 'indices'] é um mecanismo que permite pesquisar rapidamente por linhas em uma tabela, dado o valor de uma determinada coluna (ou algumas colunas) da tabela. Um *índice primário* ou *chave primária* define um valor único, que não pode ser repetido em outras linhas da tabela.
Uma *consulta* [query] é um pedido de pesquisa no banco de dados, que permite obter todo um subconjunto da tabela ou de várias tabelas, especificando as condições de seleção.

Desktop x Cliente/Servidor

Uma aplicação que utiliza bancos de dados é composta de três partes:
• *Interface com o usuário:* responsável por validar as entradas do usuário, e iniciar pesquisas de acordo com um pedido do usuário.
• *Mecanismo de acesso* [database engine]*:* responsável pela manutenção das estruturas de dados necessárias em arquivos, pelos detalhes internos do acesso aos dados, e pela manutenção da integridade dos dados.
• *Armazenamento de dados:* arquivos que contém os dados em si.
Um banco de dados "desktop" (ou baseado em arquivos) é aquele no qual a interface com o usuário e o mecanismo de acesso ficam no mesmo computador (a estação) e apenas os arquivos de dados ficam num servidor de rede. Operações de consulta ou pesquisa devem passar através da rede.
Por exemplo, quando um usuário quer ver uma relação de contas a pagar, mas apenas em determinado período, o sistema deve selecionar alguns registros baseado na data informada. No ambiente desktop, a estação traz todos os registros através da rede, mesmo os que não são utilizados. O tráfego gerado na rede é grande, principalmente quando várias estações acessam simultaneamente o servidor.
Já num banco de dados cliente/servidor, a interface com o usuário fica na estação e se comunica remotamente com o mecanismo de acesso, que é um *sistema gerenciador de banco de dados* (SGBD) rodando no servidor. Quando o SGBD recebe um pedido para selecionar alguns dados, ele acessa localmente os dados no servidor e retorna apenas o resultado pedido. No caso de uma atualização, não é necessário nem mesmo retornar um resultado, apenas informar que a atualização foi feita.

O diagrama abaixo resume as diferenças entre os ambientes:

Desktop cliente/servidor



O SQL Server, como já foi dito, é um sistema de gerenciamento de bancos de dados cliente/servidor.

2 - Instalação e Configuração

Objetivos:
- Saber o que é necessário para instalar o SQL Server em um computador;
- Aprender a instalar o SQL Server em um servidor Windows NT e a configurar as estações de rede para utilizá-lo.

Requisitos de Sistema

Antes de instalar o SQL Server, é preciso saber quais os requisitos mínimos e recomendados para a instalação.
Computador: Intel e sistemas compatíveis, ou DEC Alpha e compatíveis:
É recomendável que todos os componentes de hardware escolhidos estejam listados na HCL (lista de compatibilidade de hardware) do Windows NT.
Memória: Mínimo de 32 Mb. Recomendável memória adcional, especialmente se o servidor já estiver processando outras funções além de banco de dados, ou se forem usados bancos de dados grandes e replicação
Sistema Operacional: O SQL Server pode ser instalado no Windows NT 4.0 ou superior, com o Service Pack 3 ou posterior, nas plataformas de hardware citadas acima, ou no Windows 9x (no NT ele roda como um serviço e no 9x como uma aplicação). O software de cliente, para acesso ao SQL Server nas estações, pode ser instalado em Windows NT Server, Windows NT Workstation, Windows 95/98, ou Windows 16-bits (3.x), MS-DOS, UNIX, Macintosh, ou navegadores Internet.
Espaço em disco: Numa instalação mínima, são usados 70 MB, e numa instalação completa, 160 MB, incluindo todos arquivos de programas, documentação online, ferramentas de desenvolvimento, e arquivos de exemplo. Uma instalação de um novo servidor, só com as ferramentas de gerenciamento, exige 70 MB
Software de rede: Numa rede Windows NT, o SQL Server usa o software de rede integrado. Não é necessário software adicional, exceto para conectar a alguns outros tipos de rede. No caso da Novell Netware, o suporte é fornecido pelo protocolo NWLink do Windows NT.
Não é recomendado instalar o SQL Server num computador que seja controlador primário de domínio (PDC) ou controlador reserva (BDC).

Outras considerações

Sistemas de arquivos: o Windows NT suporta FAT ou NTFS. O SQL Server pode usar qualquer um dos dois indiferentemente e o tipo de sistema usado não afeta seu desempenho (exceto quando se usa compressão no NTFS, que reduz o desempenho). Recomenda-se o NTFS para a instalação do servidor, pelas suas vantagens de recuperação e segurança..

Nome do Servidor: o nome do computador usado para o SQL Server deve seguir certas regras:

- Nomes não devem ter mais de 128 caracteres.
- Nomes não podem conter espaços.
- Nomes devem começar com uma letra (maiúscula ou minúscula) ou sublinhado ('_'). Nomes também podem começar com @ ou #, mas como nomes começados por essas letras têm um significado especial, não serão usados na maioria dos casos.
- Caracteres no nome depois da primeira letra, podem ser qualquer letra, número, ou os símbolos @, _, #, ou $.

Opções Usadas na Instalação

Informação de Registro

Nessa fase, informe seu nome, nome da empresa e número de identificação do produto.

Escolha do tipo da instalação

O SQL Server oferece três tipos de instalação: Típica (Typical), Mínima (Minimal), e Personalizada (Custom).

Instalação típica

Automaticamente instala o SQL Server e utilitários de cliente com as opções padrão de instalação. São instalados: SQL Server, Book Online, Quick Tour, e What's New. Essa oção exige cerca de 163 MB de espaço em disco, e não instala o softwarede Full Text Search, as ferramentas de desenvolvimento ou os arquivos de exemplo.

Instalação Compacta

Essa opção instala o mínimo de arquivos necessários para operar o SQL Server, e usa todas as opções padrão de instalação. Uma instalação compacta é como uma instalação típica, exceto que as ferramentas de gerenciamento, os livros on-line, Quick Tour e What's New não são instalados. Esta opção exige cerca de 74 MB de espaço em disco.

Instalação Personalizada

Essa opção permite fazer a escolha de quais componentes exatamente serão instalados, além de permitir escolher muitas opções diferentes do SQL Server, ao invés de apenas aceitar automaticamente as opções padrão.

Instalação remota

É possível instalar o SQL Server remotamente em outro computador. Nesse caso, você pode informar o nome do computador e as localizações do Windows NT e da unidade onde o SQL Server será instalado.

Diretórios

O SQL Server é instalado por default no diretório C:\MSSQL, mas isso pode ser alterado. O nome do diretório pode ser longo, mas não deve conter espaços. Dentro desse diretório, todos os arquivos possuem nomes curtos (FAT 8.3), mesmo num drive que suporta nomes longos. Nessa apostila, ao fazermos referência a *C:\MSSQL*, note que você pode ter instalado em um diretório diferente. Nesse caso, substitua o nome pelo do seu diretório.
O SQL cria os seguintes subdiretórios durante a instalação:
• BACKUP - Contém arquivos de backup.

• BINN - Contém arquivos executáveis das ferramentas de administração do SQL e arquivos do Help Online, e DLLs.
• FTDATA - Só é instalado se o componente de procura por texto completo [Full-Text Search] tiver sido instalado. Utilizada para lidar com arquivos de catálogo somente de texto.
• DEVTOOLS - Essa pasta, e suas subpastas, só são criadas, se escolher-se instalar as ferramentas opcionais de desenvolvimento e exemplos.
• HTML - Armazena arquivos HTML e arquivos relacionados.
• DATA - Contém os arquivos de dados.
• JOBS - Armazena informações sobre tarefas [Jobs] do SQL Server
• INSTALL - Scripts de instalação e arquivos de saídas.
• LOG - Contém arquivos de log de erro.
• REPLDATA - Diretório de trabalho usado para replicação.
• UPGRADE - Arquivos do assistente de atualização de versão. Não é criada se você não instalar o assistente de atualização.

Conjunto de caracteres, ordem de classificação, e comparação Unicode

O *conjunto de caracteres* [character set] usado determina os tipos de caractere que o SQL reconhece nos seus dados e a *ordem de classificação* [sort order] determina como o SQL Server compara dados em forma de caractere e como eles são classificados. Você deve escolher essas opções durante a instalação. A única forma de alterá-las posteriormente é reinstalando o SQL Server.
Como geralmente são usadas aplicações Windows para acesso aos dados, é altamente recomendável usar o conjunto de caracteres ISO 8859-1 (Code Page 1252), também chamado Latin-1 ou "ANSI". Esses caracteres são os mesmos usados no Windows. (Esse é o conjunto de caracteres default durante a instalação). Existe outros caracteres como: 850(multilingual) que inclui todos os caracteres da Europa,América do Norte e América do Sul , 437(US English) que contém o alfabeto completo do Estados Unidos, 932(Japanese) contém o alfabeto completo do Japão.
A ordem de classificação determina, por exemplo, se o SQL Server considera ou não as letras acentuadas (ç, é, ã) como diferentes dos caracteres não acentuados. Para a língua portuguesa, é recomendável usar a ordem "Dictionary order, case-insensitive, accent-insensitive" [ordem de dicionário, insensível ao caso, insensível a acentos]. Dessa forma, caracteres acentuados e não-acentuados são tratados da mesma forma, como também letras maiúsculas e minúsculas. Por exemplo, ao pesquisar "CAMARA" no banco de dados, "Câmara" será considerado igual. (Mas 'c' e 'ç', ou 'C' e 'Ç' são diferentes). Existem outras opções para ordem de classificação como: Dictionary order,case-insensitive [ordem de dicionário insensível ao caso] , neste caso as letras maiúsculas são tratadas da mesma forma que as letras minúsculas, mas os caracteres acentuados são tratados de forma diferente dos caracteres não acentuados (essa é a ordem de classificação default durante a instalação).
O SQL Server tem a capacidade de armazenar caracteres ASCII padrão, e caracteres Unicode. Os caracteres Unicode são capazes de representar mais de 64000 caracteres diferentes., embora os caracteres ASCII sejam capazes de representar apenas 256 caracteres. Depois de escolher o conjunto de caracteres e a ordem de classificação, você deve escolher uma comparação Unicode [Unicode collation], que funciona como uma ordem de classificação para os caracteres Unicode armazenados no SQL Server.
Uma comparação Unicode consiste de um local e vários estilos de comparação. Locais, normalmente nomeados de por países ou regiões culturais, ordenam caracteres de acordo com o padrão naquela área. O programa de instalação do SQL Server vai fornecer uma comparação Unicode padrão, com base no conjunto de caracteres e ordem de classificação que você

escolheu. É recomendável que não se altere essa seleção, pois caso ela seja alterada, a migração de Unicode para não-Unicode torna-se mais difícil, e dados Unicode e não-Unicode podem ser ordenados de maneiras diferentes.

Protocolos de Rede

Para cada tipo de cliente de rede, o SQL Server possui uma *Net-library*, um driver que suporta comunicação através desse tipo de rede. As opções disponíveis são:
• *Named Pipes (Netbeui):* Suporta o protocolo Netbeui. É instalado por default. Você não deve removê-lo pois os utilitários do SQL Server depende desta Net-Library.
• *Multi-protocol:* suporta a conexão através de vários protocolos de rede, suportando também segurança integrada e criptografia (caso a aplicação suporte).
• *NWLink IPX/SPX:* comunica-se com clientes Netware. O SQL Server pode se registrar como um serviço numa rede Netware.
• *TCP/IP sockets:* suporta comunicação através de Windows sockets, por exemplo, com um cliente de Internet.
• *Banyan VINES, AppleTalk ADSP, DECnet:* outros tipos de rede.

Opções de auto-inicialização

Serviço é um programa executável que não tem interface com o usuário, mas tem formas de controlar. Como iniciar e parar o serviço determinado.
Definem se os serviços MSSQLServer e SQLServerAgent iniciarão automaticamente com o Windows NT ou serão iniciados manualmente. Essa opção pode ser alterada depois no Painel de Controle [Control Panel] do Windows NT, opção Serviços [Services].
Para o serviço SQLServerAgent, você pode também definir qual o nome de usuário e senha que ele utiliza para se conectar ao sistema.

Modo de licenciamento

O SQL Server pode ser licenciado *por servidor* [per server], onde para cada servidor adquire-se N licenças de acesso, ou *por estação* [per seat], onde existe uma licença para cada estação, independentemente do número de servidores usados. Durante a instalação você pode definir qual dos modos utilizar. Para alterar essas opções posteriormente, use o ícone Licenciamento [Licensing] no Painel de Controle do Windows NT.

Instalando o software de servidor

Criando uma conta para o SQLServerAgent

Durante a instalação, SETUP pede um nome de conta de usuário e a senha dessa conta, para uso do serviço SQLExecutive. Existem duas maneiras de selecionar o usuário :
• Criar uma conta no Windows NT .
• Utilizar a conta Local System do Windows NT. Mas nesse caso nem todos os recursos do SQLExecutive podem ser usados. Por exemplo:
Ao utilizar Backups poderá não se ter acesso ao driver de destino do Banco de Dados.
É recomendável criar uma nova conta de usuário antes de instalar. Essa conta não deve ser usada por usuários para logar no computador. Ela será exclusiva para o SQL Executive.
Para isso, abra o Gerenciador de Usuários [User Manager] do Windows NT, no grupo/menu de programas Ferramentas Administrativas [Administrative Tools]. Clique em **Usuário|Novo usuário...** [User|New user...]. Informe o nome de usuário "SQLExecutive" (o nome pode ser qualquer). Informe uma senha e anote-a para usar mais tarde com o SETUP. Desmarque a

opção "O usuário deve alterar a senha no próximo logon" [The user must change password...] e marque "A senha nunca expira" [Password never expires].
Depois acrescente esse usuário ao grupo local "Administradores" [Administrators]. Use o menu **Diretivas|Direitos de usuário...** [Policies|User rights] para conceder os direitos de "Logon como serviço" [Logon as a service], "Agir como parte do sistema operacional" [Act as part of the operating system], "Aumentar cotas" [Increase quotas], "Substituir um token de nível de processo" [Replace a process level token] à conta 'SQLExecutive' . Como são direitos avançados, marque a opção Exibir direitos avançados do usuário [Advanced User Rights].

Iniciando o SETUP

Para instalar o SQL Server 7, efetue logon no Windows NT com uma conta que tenha privilégios administrativos, coloque o CD de instalação na unidade de CD . Em alguns segundos, o programa de Instalação do SQL Server irá iniciar automaticamente e mostrar a tela abaixo:

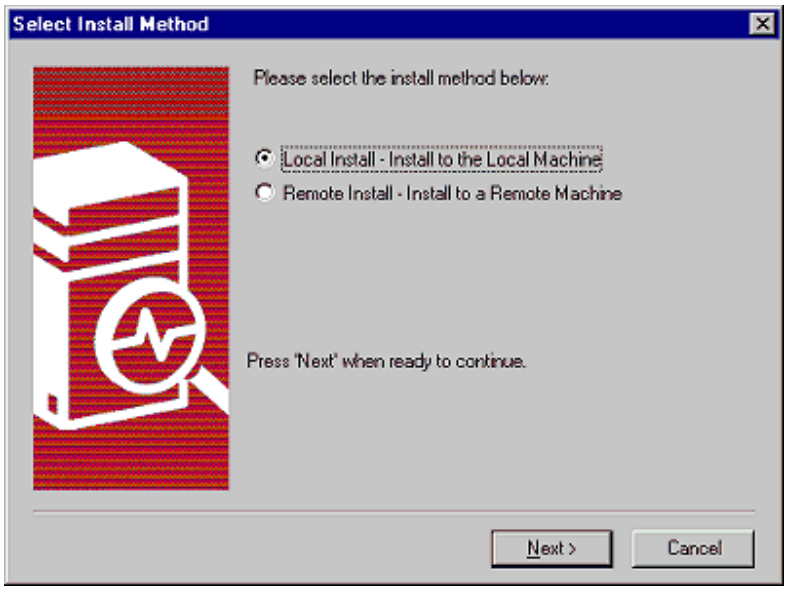
Nessa tela você tem diversas opções. Para iniciar a instalação do SQL Server, selecione "Install SQL Server 7.0 Components". A próxima tela permite que se selecione quais componentes se quer instalar.
Nota: Se você tiver a opção de autoReprodução do CD desativada, você pode iniciar manualmente o programa de instalação do SQL Server. Para fazê-lo, vá para o diretório raiz do CD de instalação do SQL Server, e execute o programa "setup.bat". Então será mostrada a tela acima. Se o SQL Server estiver sendo instalado de um compartilhamento da rede, mude para a pasta do compartilhamento que contem o SQL Server e execute o arquivo "setup.bat".

Nota: A opção "Install SQL Server Prerequisites" permite intalar os softwares necessários para se poder instalar o SQL Server. Estes são o service pack 4 do Windows NT e o Internet Explorer 4.01.

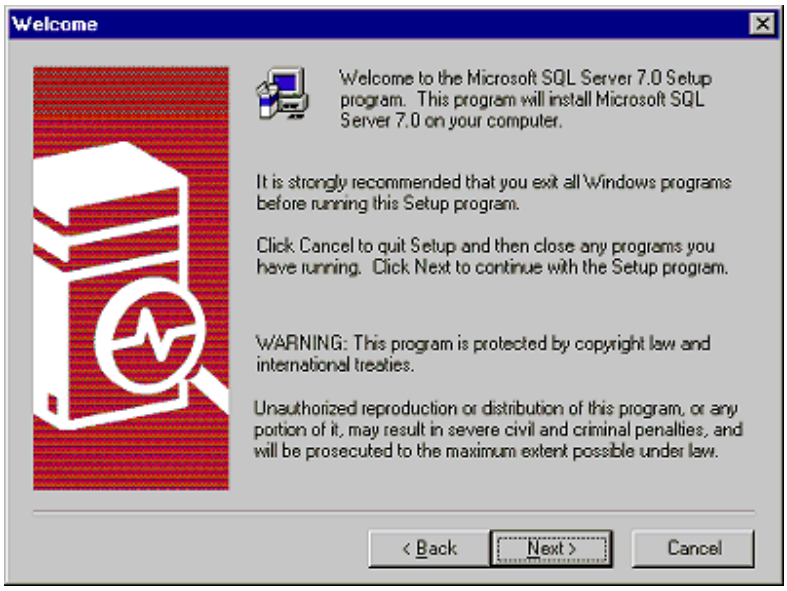
Install SQL Server 7.0 Components

Nessa tela, selecione "Database Server- Standard Edition" se você quiser instalar o SQL Server no Windows NT, ou "Database Server- Desktop Edition" se você quiser instalar o SQL Server no windows 9x.

Select Install Method



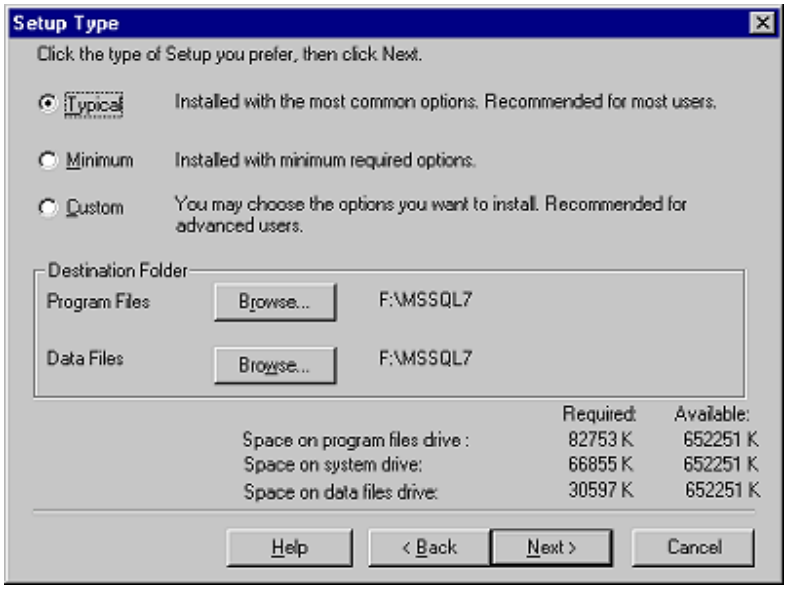
Se você quisesse instalar em um computador remoto, clicaria em "Remote Install" e informaria as opções do computador remoto. No nosso caso, clique em Next para fazer a instalação local. Aí, aparece a tela de boasa-vindas:



Clique em Next para continuar.
Aparece a tela de "Acordo de Licença de Software" [Software License Agreement]. Se você aceita os termos do acordo, clique em Yes para continuar. Você deve selecionar Yes se você quer instalar o SQL Server.
Aí aparece a tela de informação do usuário. Aqui, entre seu nome e o nome da companhia. Depois que tiver entrado com essas informações, clique em Next para continuar.

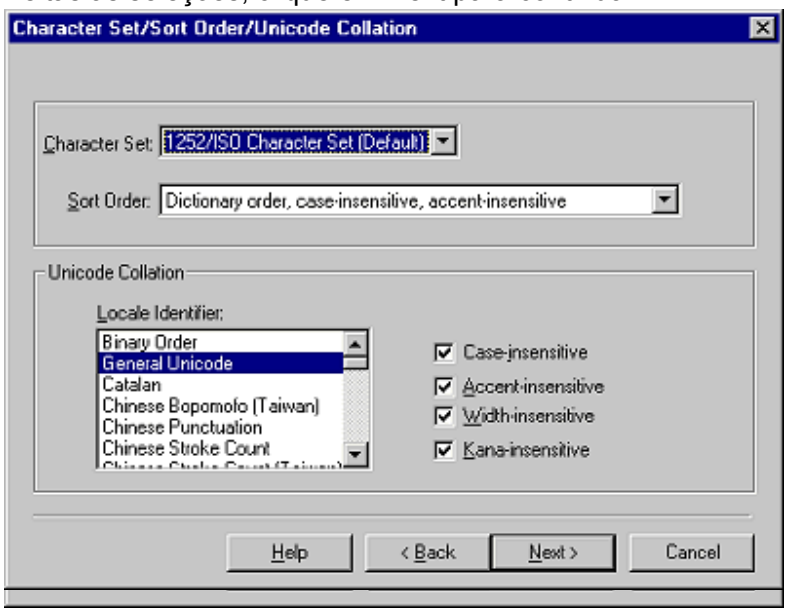
A seguir será pedido o número de série do SQL Server. Este pode ser encontrado no adesivo amarelo colado na caixa do CD. Depois de entrar com esse número, clique em Next para continuar.

Setup Type



Aqui, você deve selecionar se você quer fazer uma instalação mínima, típica, ou personalizada. Já discutimos sobre cada uma das opções anteriormente. Aqui, usaremos a instalação personalizada, já que ela fornece a maior flexibilidade quando instalano o SQL Server. Clique no botão perto de "Custom" para escolher a instalação personalizada.
Ainda nesta janela, você deve decidir onde armazenar os arquivos de programa e de dados do SQL Server. Podem ser instalados no mesmo local, ou em locais diferentes. Use os botões "Browse" para selecionar outros locais que não sejan os locais padrão selecionados. Use as informações de espaço exigido [Required] e disponível [Available] para decidir melhor onde instalar os arquivos.

Feitas as seleções, clique em Next para continuar.



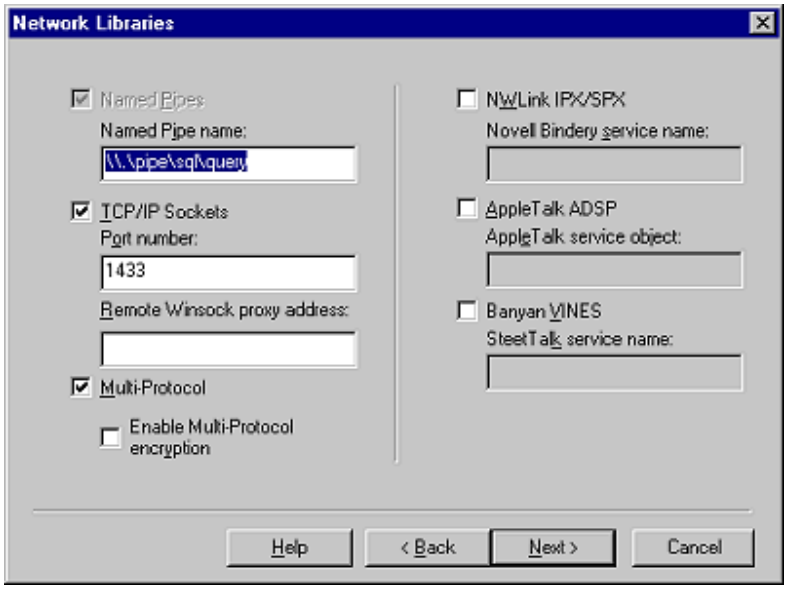
Select Components

Aqui, você deve escolher aqueles componentes do SQL Server que você quer carregar. Perceba que no lado esquerdo da tela estão os componentes, e no lado direito os sub-componentes. Primeiro, você seleciona um componente clicando dentro da caixa de verificação, e se houver subcomponentes, você os escolhe clicando em caixas de verificação do lado direito. Enquanto você escolhe os componentes para sua instalação, perceba que você pode determinar o tamanho da instalação observando o espaço disponível em disco na parte inferior da janela. Depois de escolhido o que for apropriado, clique em Next para continuar.

Character Set/ Sort Order/ Unicode Collation

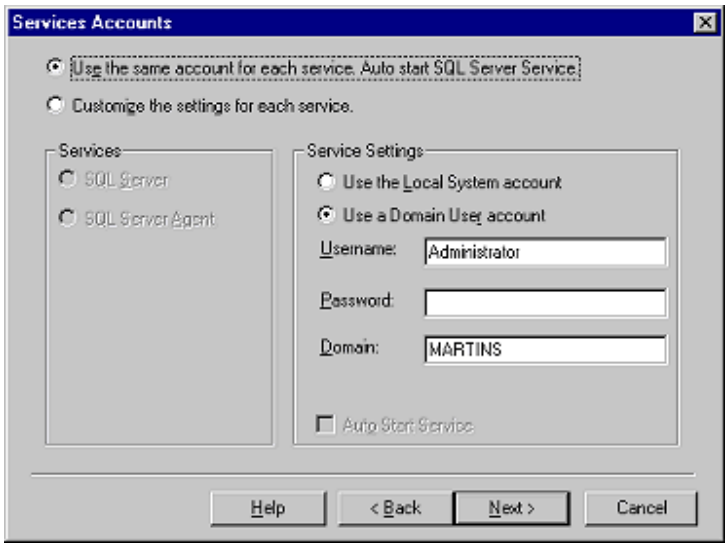
Aqui você escolhe tanto o conjunto de caracteres [Character Set] quanto o ordem de classificação [Sort Order] da caixa de lista correspondente no topo da tela. Conforme já discutido, recomenda-se deixar o padrão para o "Character Set" e pôr a "Sort Order" como "Dictionary order, case-insensitive, accent-insensitive".
Como já citado, a não ser que você tenhas uma razão realmente boa para mudar essa opção, deixe a opção padrão selecionada. Depois de terminar as suas escolhas, clique em Next para continuar.
Considerações importantes: Um conjunto de caracteres pode ser modificado depois da instalação, mas exige que você reconstrua todos seus bancos de dados e recarregue os dados (uma tarefa dispendiosa!). Também é recomendável que todos as instalações do SQL Server que precisem comunicar-se usem o mesmo conjunto de caracteres,e ordem de classificação, ou você poderá ter resultados inesperados.

Network Libraries



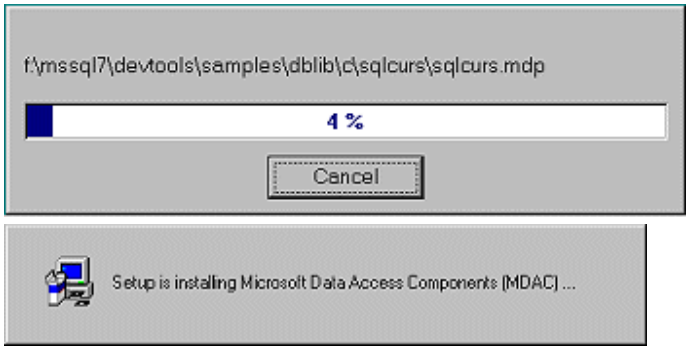
Nessa janela, de seleção das bibliotecas de rede [Network Libraries], você deve, para cada biblioteca de rede que for instalar, entrar com alguns parâmetros adicionais, como a portão em que o servidor vai escutar, para a biblioteca de rede TCP/IP, e outras de acordo com a biblioteca de rede a ser instalada. Como já foi dito, a biblioteca de rede Named Pipes deve ser selecionada, pois é utilizada durante a instalação. Depois da instalação concluída, ela pode ser retirada, apesar de não ser recomendável.
Normalmente, as opções padrão funcionam bem. Depois de feitas as seleções necessárias, clique em Next para continuar.
Nota: Para mudar uma net-Library depois da isntalação, use o "SQL Server Network Utility" (SRVNETCN.EXE)

Services Accounts



Aqui você pode usar a mesma conta para os dois serviços (SQLServer e SQLServerAgent), iniciando automaticamente o serviço SQL Server. Para isso, selecione o primeiro botão [Use the same account for....]. Caso você queira usar uma conta diferente para cada serviço, selecione "Customize the settings for each service" e os botões SQL Server e SQL Server Agent se tornarão ativos. Aí as opções do serviço [Service Settings] serão referentes ao serviço selecionado.
Se foi criada a conta do usuário para uso do SQLServerAgent, informe o nome do usuário, a senha e confirme a senha (o default é o nome de usuário conectado atualmente). Informe também o domínio do NT Server onde essa conta existe. Caso não tenha criado a conta marque a opção Install to log on as Local System account (é uma conta predefinida do Windows NT).
Nota: Caso se escolha usar a conta de sistema local, o SQL Server não será capaz de se comunicar com outros servidores.
Clique em Next para continuar.
A seguir aparece a tela de seleção do modo de licenciamento [Choose Licensing Mode]. Escolha o licenciamento Per Server ou Per Seat, levando em conta o que foi discutido anteriormente. Clique em Next para continuar.

"Copying Files" e "SETUP is..."



...

O SETUP vai copiar os arquivos necessários para o diretório de instalação, reindexar as tabelas de sistema, e definir a configuração inicial do SQL Server. Esse processo leva cerca de 10-15 minutos, dependendo da velocidade do seu computador.
Após esse processo, o SQL Server terá sido instalado e estará pronto para usar. As ferramentas do SQL Server estarão disponíveis no submenu "Microsoft SQL Server 7.0", dentro do menu Iniciar|Programas [Start|Programs] do Windows NT 4.0. Você precisará de iniciar seus serviços antes de poder conectar-se pela primeira vez ao servidor SQL Server.

Depois que o SQL Server estiver instalado, há alguns passos a serem completados antes de se poder dizer que o SQL Server está pronto para rodar. Alguns desses passos devem ser executados apenas uma vez para se assegurar que o SQL Server foi instalado corretamente e para prepará-lo para rodar corretamente no futuro. Aqui vamos citar esses passos. Outros

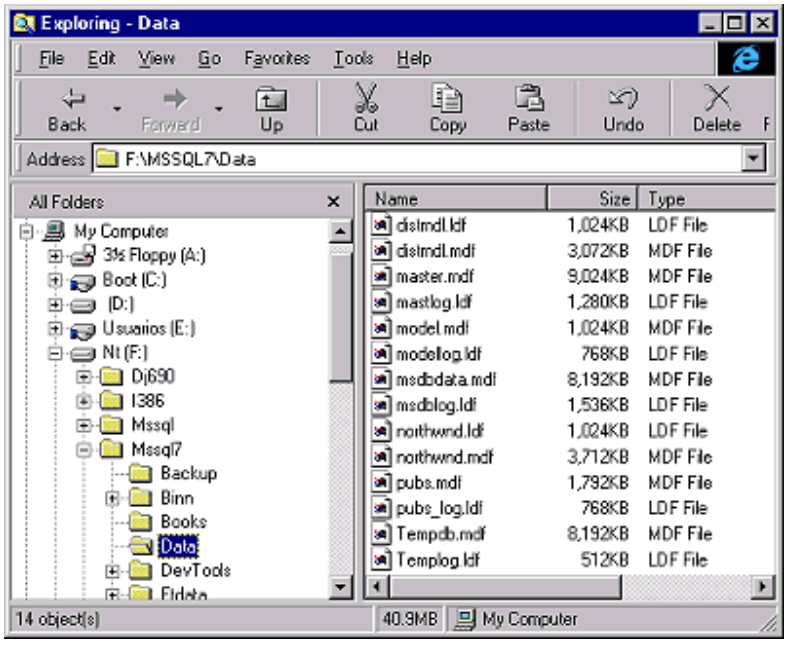
passos, que incluem configurações, ajustes e otimização, estabelecimento de IDs para login e IDs de usuários do banco de dados, e é claro, a criação de novos bancos de dados, serão discutidos em outras seções.
Nesta seção o objetivo será verificar se o SQL Server foi instalado corretamente.

Verificando se o SQL Server foi instalado corretamente

Depois que o SQL Server estiver instalado, você deve gastar alguns minutos para verificar que o mesmo foi instalado e está rodando adequadamente. Use os passos abaixo para verificar a instalação.

Verifique se as pastas de programas e dados foram criadas

Use o NT Explorer (ou Windows Explorer) para verificar que os arquivos de programas do SQL Server e os arquivos de dados foram instalados nas pastas que você especificou durante o processo de instalação.
Cada pasta deve conter pastas adicionais de acordo com o que foi dito anteriormente (em Diretórios). Verifique se todas as subpastas tanto nas pastas de programas quanto de arquivos existem.
Deve-se verificar o conteúdo da subpasta /data, que está abaixo da pasta de dados que você especificou. É aí que o SQL Server cria e armazena vários bancos de dados padrão e logs de transações. A pasta /data deve se parecer com a figura abaixo.



Os arquivos são:

- Distmdl.mdf e Distmdl.ldf (apenas se obejetos opcionais de replicação foream instalados.
- Northwnd.mdf e Northwnd.ldf

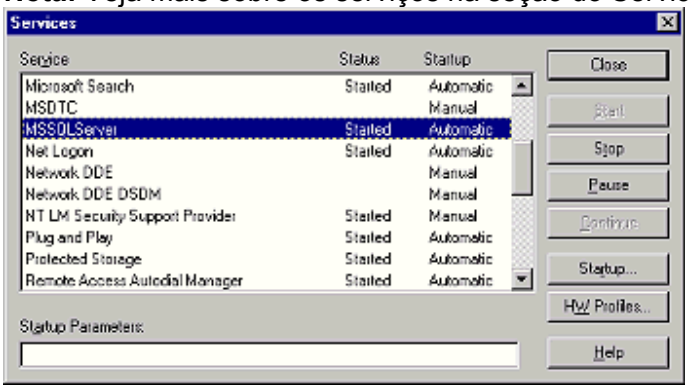
- Master.mdf e Mastlogldf
- Model.mdf e Modellog.ldf
- Msdbsdata.mdf e Msdbblog.ldf
- Pubs.mdf e Pubs_log.ldf
- Tempdb.ldf e Templog.ldf

Verificar que todas as ferramentas de gerenciamento do SQL Server estão instaladas

Verifique se as ferramentas de gerenciamento do SQL Server que você especificou durante a instalação foram instaladas. O modo mais fácil de fazer isso é clicando em Iniciar | Programas | Microsoft SQL Sever 7.0. Aí você verá um menu com a listagem de todos os programas, utilitários e documentação online que você especificou durante a instalação.

Verifique que os serviços do SQL Server estão carregados e executando

Quando você instalou o SQL Server, seus dois serviços principais - SQLServerAgent e MSSQLServer - foram instalados e configurados para iniciar automaticamente (assumindo que você quis inicialização automática). Quando o SQL Server é instalado pela primeira vez, os dois serviços não são iniciados automaticamente até que o NT Server seja reiniciado.
Nota: Veja mais sobre os serviços na seção do Service Manager.



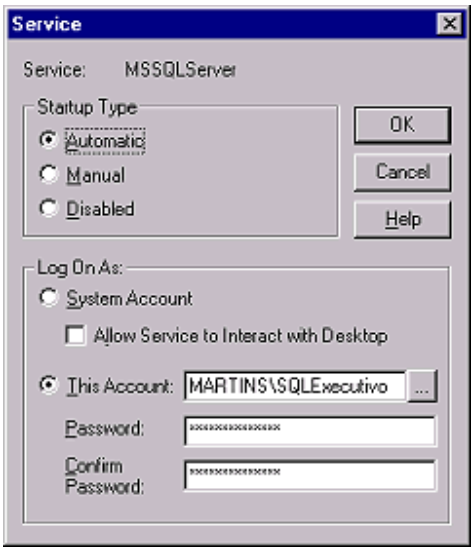
Conferindo se os dois serviços foram instalados, e manualmente iniciando-os pela primeira vez

1. Abra o Painel de Controle do NT, e então dê um duplo clique no ícone Serviços. Aparece então a caixa de diálogo de Serviços.
2. Percorra a lista de serviços até que você veja pela primeira vez o serviço MSSQLServer. Se você não reinicializou o computador desde que instalou o SQL Server, ele deve ter um Estado [Status] em branco. E em Inicialização [Startup], deve estar configurado como automático.
3. Para iniciar manualmente o serviço, clique nele uma vez para que o mesmo fique selecionado, e então clique no botão Iniciar [Start]. Depois de uns 30 segundos, o Estado [Status] vai mudar para Iniciado [Started] e o serviço está agora sendo executado.
4. Percorra a lista um pouco mais até que você veja o serviço SQLServerAgent, e ele também deve estar com o Estado em branco e a Inicialização como Automática.
5. Para iniciá-lo manualmente, faça da mesma maneira que foi feito com o serviço MSSQLServer, mas agora selecionando o serviço SQLServerAgent. Depois de uns 15

segundos, o Estado [Status] vai mudar para Iniciado [Started] e o serviço está agora sendo executado.

6. Se tudo estiver correto, feche a caixa de diálogo de Serviços e o Painel de Controle.

E se os serviços não estiverem aí? Se os serviços não estiverem listados, sua instalação do SQL Server falhou e você vai ter que reinstalá-lo, mas isso é raro.
Você pode ter notado que na coluna Inicialização [Startup], estava a a palavra Manual, ao invés de Automático. Se você se vir nessa situação, você provavelmente esqueceu de escolher a opção auto-iniciar [Auto-start] quando você instalou o SQL Server. É bem fácil corrigir isso.



Escolhendo auto-iniciar

1. Um de cada vez, selecione cada um dos serviços do SQL Server na caixa de diálogo de Serviços e clique no botão Inicialização [Startup]. Isso mostra a janela abaixo.
2. Nessa janela, selecione Automático como o tipo de Inicialização [Startup type].
3. Clique em OK, e você retornará para a caixa de diálogo de Serviços.
4. Repita os passos 1 a 3 para o outro serviço do SQL Server.
5. O último passo é iniciar manualmente os serviços como descrito anteriormente. Se os serviços não iniciarem conforme descrito, e se você receber uma mensagem de erro, veja a solução de problemas de instalação para tentar resolver o problema.

Verifique que você consegue se logar no SQL Server

A última maneira de verificar que o SQL Server foi instalado corretamente é tentar se logar e executar uma pequena consulta. Se você puder executar essas duas tarefas com sucesso, você saberá que o SQL Server foi instalado sem problemas.
Aqui vamos demonstrar como se logar ao servidor usando a ferramenta ISQL, que é o programa baseado em linha de comando fornecido com o SQL Server, utilizado para executar comandos Transact-SQL. Embora haja outros programas ou utilitários do SQL Server que você poderia usar ao invés do ISQL, ele é recomendado para este teste porque há menos coisas para dar errado quando da execução do teste.
Execute e se logue ao ISQL, e rode uma pequena consulta para verificar que o SQL Server foi instalado corretamente.

1. Vá para o prompt de comando do NT.
2. No prompt de comando, escreva o seguinte e pressione Enter:

```
isql /Usa /P
```

3. Se tudo estiver funcionando corretamente, o prompt do ISQL deverá aparecer ao invés do prompt de comando. Ele se parece com isso:
   ```
   1>
   ```
4. Escreva a seguinte consulta para testar se o SQL Server vai responder. Pressione Enter depois de cada linha.
   ```
   select @@servername
   go
   ```
5. Assumindo que tudo esteja funcionando, o nome do seu servidor deve ser mostrado no prompt do ISQL. A resposta deve se parecer (o número e nome do servidor vai variar) com isso:
   ```
   PDC
   (1 row affected)
   1>
   ```
6. Para sair do programa ISQL, digite **exit** no prompt do ISQL e pressione Enter, e você retornará para o prompt de comando do NT.

Se tudo tiver ocorrido como descrito, você sabe que o SQL Server foi instalado corretamente e está funcionando adequadamente. Se você encontrar problemas ou mensagens de erro, veja a solução de problemas de instalação para tentar resolver o problema.

Instalando o software de cliente

Quando você já tiver instalado o SQL Server, é hora de pensar em instalar as ferramentas de gerenciamento nas estações de trabalho que irão ser utilizadas para administrar remotamente o SQL Server. Também pode ser interessante instalar as ferramentas de gerenciamento nas estações de trabalho dos desenvolvedores.
As ferramentas de gerenciameno são as mesmas que você teve a chance de instalar quando da instalação do SQL Server (afinal, mesmo na máquina onde o servidor está instalado, vocvê precisa de software de cliente para conectar ao servidor). A vantagem de instalar as ferramentas de gerenciamento em outras máquinas é que você pode gerenciar remotamente o SQL Server de virtualmente todos computadores, não apenas do servidor físico em que o SQL Server está sendo executado.
As ferramentas de gerenciamento do SQL Server podem ser instaladas no NT Server, NT Workstation, e Windows 9x. Ao contrário de algumas das ferramentas de gerenciamento incluídas no SQL Server 6.5, as ferramentas de gerenciamento do sQL Server 7.0 não podem ser executadas sob o Windows 3.x ou DOS.
Aqui descreveremos as ferramentas de gerenciamento, como instalar e utilizá-las. Todas as ferramentas de cliente aqui descritas são as mesmas descritas em Ferramentas de gerenciamento, mas algumas delas, especificamente o Service Manager e o Server Network Utility só são instaladas no servidor. Onde se vir escrito "Ferramentas de gerenciamento" nesta seção, subentende-se que se está citando as ferramentas de cliente (que nada mais são do que as ferramentas de gerenciamento instaladas em um cliente).
Nota: Qualquer das ferramentas de gerenciamento só podem ser utilizadas por um usuário que tenha as permissões necessárias para tal tarefa.

Ferramentas de cliente

O SQL Server inclui uma porção de ferramentas de administração para serem instaladas no cliente, que podem ser usadas para gerenciar o SQL Server. Durante o processo de instalação, você pode instalar quantas ferramentas de gerenciamento você achar necessário. O SQL Server oferece estas opções:

- **SQL Server Enterprise Manager:** Se você quer administrar remotamente o SQL Server, então a instalação do Enterprise Manager é necessária. Este programa permite ao DBA executar virtualmente qualquer tarefa administrativa no SQL Server.
- **SQL Server Profiler:** Esta ferramenta é usada para monitorar e registrar a atividade dos bancos de dados entre o SQL Server e os clientes. Apenas instale esta ferramenta nas máquinas que executarão esta tarefa.
- **SQL Server Query Analyzer:** Esta ferramenta é utilizada para enviar manualmente comandos Transact-SQL e procedimentos armazenados para o mecanismo de banco de dados do SQL Server. Você provavelmente instalará esta ferramenta na maioria, talvez em todas as máquinas de gerenciamento remoto.
- **Client Diagnostic Utilities:** Esta ferramenta é utilizada para verificar qual DB-Library está instalad em um cliente, e para configurar as ferramentas de gerenciamento para se comunicar com o SQL Server em uma rede.
- **MS DTC Client Support:** A ferramenta de Coordenação de Transações Distribuídas (DTC) da Microsoft fornece suporte aos clientes DTC. Apenas necessita ser instalada em clientes que executem uma aplicação do SQL Server que exigam o DTC.
- **Development files:** Esses arquivos são necessários para desenvolvedores OLE-DB para a criação de programas utilizando ODBC, DB-Library, ODS, SQL-DMO, Embedded SQL for C, e MS DTC. Geralmente, apenas desenvolvedores SQL Server precisarão desses arquivos.
- **Sample files:** Estes arquivos são arquivos de exemplo feitos para os desenvolvedores analisarem e aprenderem com eles. Assim como os arquivos de desnvolvimento, apenas desenvolvedores precisarão deles.
- **Replication Conflict Resolution Tool:** Utilizada para ajudar a resolver conflitos de replicação entre dois servidores SQL Server. Você apenas precisa desta ferramenta se você implementar replicação em seus servidores.
- **Livros online:** Esta é uma documentação completa e comprensiva do SQL Server, e é interessante instalá-la em todo cliente. Ocupa cerca de 15 MB de espaço no disco, mas vale a pena. Se você não quiser ocupar todo esse espaço em um cliente, você também pode instalar os livros online em um compartilhamento de rede, e então conectar a ele quando necessário. Você também tem a opção de executá-lo a partir de um drive de CD instalado localmente.

Não importa quais das opções acima você decidir instalar, as seguintes ferramentas e arquivos são instalados automaticamente. eles incluem bcp, isql, osql, ODBC, e DB-Library. Falaremos deles mais tarde.

Software e Hardware necessário para a instalação do software de cliente

- **CPU:** No mínimo um Alpha AXP ou um Intel de 32 bits (80486). Recomenda-se um Pentium 200 ou mais veloz.
- **RAM:** No mínimo 32 MB. Recomenda-se 32 MB no Windows 9x, e 64 MB ou mais no Windows NT.
- **Monitor e placa de vídeo:** Qualquer placa de vídeo que esteja instalada corretamente funcionará. Recomenda-se pelo menos um monitor de 15" com resolução de 1024x768.
- **Espaço em disco:** 73 MB para a instalação de todas as ferramentas de gerenciamento. Como provavelmente não serão instaladas todas elas, pode ser suficiente menos espaço.
- **Drive de CD-ROM:** Apenas exigido se a instalação estiver sendo feita a partir de um CD.

- **Placa de rede**: Qualquer placa de rede que funcione no sistema operacional será aceita. Recomenda-se uma placa de 10 ou 100 Mbits se você estiver acessando um ou mais servidores SQL Server pesadamente a partir da estação trabalho.
- **Sistema operacional:** No mínimo NT Server ou Workstation 4.0 (com Service Pack 3), ou Windows 9x. Recomenda-se usar sempre o Service Pack mais recente em qualquer desses sistemas.

De onde instalar as ferramentas de gerenciamento

As ferramentas de gerenciamento do SQL Server podem ser instaladas a partir dos seguintes locais.

- **CD de instalação do SQL Server:** As ferramentas de grenciamento podem ser instaladas diretamente do CD executando-se o programa de instalação do SQL Server. Este é o mesmo programa utilizado para instalar o SQL Server.
- **Compartilhamento de rede**: O meio mais flexível de se instalar as ferramentas de gerenciamento é instalá-las a partir de um compartilhamento na sua rede. O processo de instalação se inicia quando se executa o arquivo setup.bat, o mesmo utilizado para instalar o SQL Server. Se você pretende isntalar diversas cópias das ferramentas de administração em várias máquinas, este é o meio mais eficiente.

Como instalar as ferramentas de gerenciamento

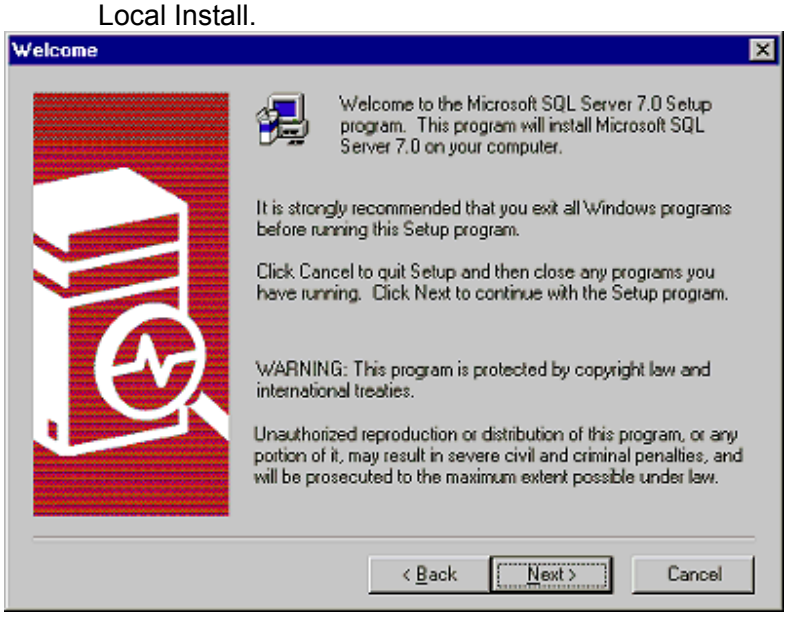
A instalação das ferramentas de gerenciamento do SQL Server é um processo simples, Na verdade, os passos exigidos para a instalação das ferramentas de gerenciamento são praticamente os mesmos necessários para instalar o SQL Server. Isso significa que você já conhece o processo. Abaixo listam-se os passos necessários para a instalação das ferramentas de gerenciamento do SQL Server em um cliente.

1. Se você estiver instalando as ferramentas de cliente no Windows NT, você deve efetuar logon com uma conta que tenha privilégios administrativos. Se você estiver instalando-as no Windows 9x, você pode efetuar logon com qualquer conta de usuário.
2. Certifique-se de que não há nenhum outro programa sendo executado, antes de iniciar o processo de instalação. Se houver, feche-os antes de continuar. Confirme também se há alguma versão antiga das ferramentas de gerenciamento instaladas na máquina. Se houver, remova-as antes de instalar a nova versão.

3. Se você estiver fazendo a instalação a partir de um CD, apenas insira o CD do SQL Server no drive de CD. Em alguns segundos, a tela do programa de instalação do SQL Server vai surgir automaticamente. Caso você não esteja instalando de um CD, execute o arquivo **setup.bat** do compartilhamento de rede onde estão os arquivos de instalação. Aparece a tela abaixo.

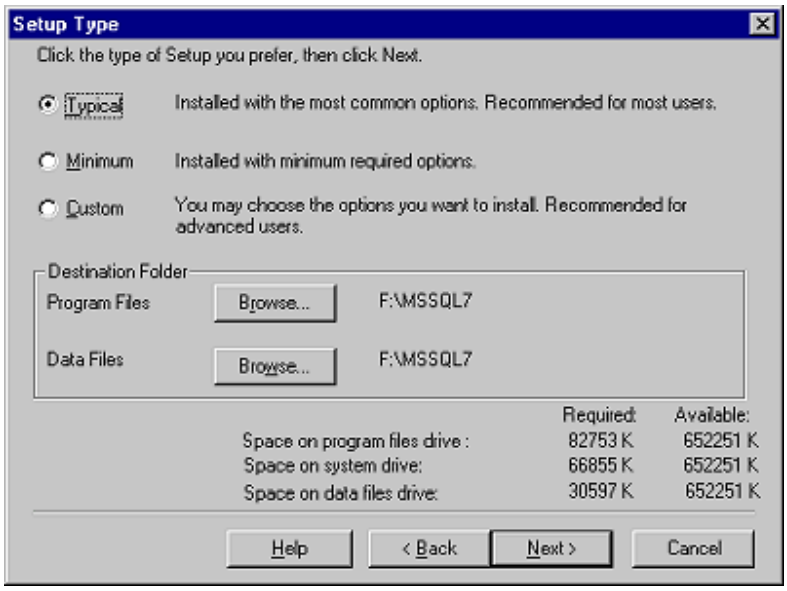
Nesta tela, você tem uma porção de opções. Selecione Install SQL Server 7.0 Components.

4. Na próxima tela, escolha Database Server- Desktop Edition.
5. Depois você deve escolher se vai fazer uma instalação para a máquina local ou para uma máquina remota. Supondo que você esteja instalando as ferramentas de gerenciamento estando fisicamente na máquina em que elas serão executadas, escolha Local Install.

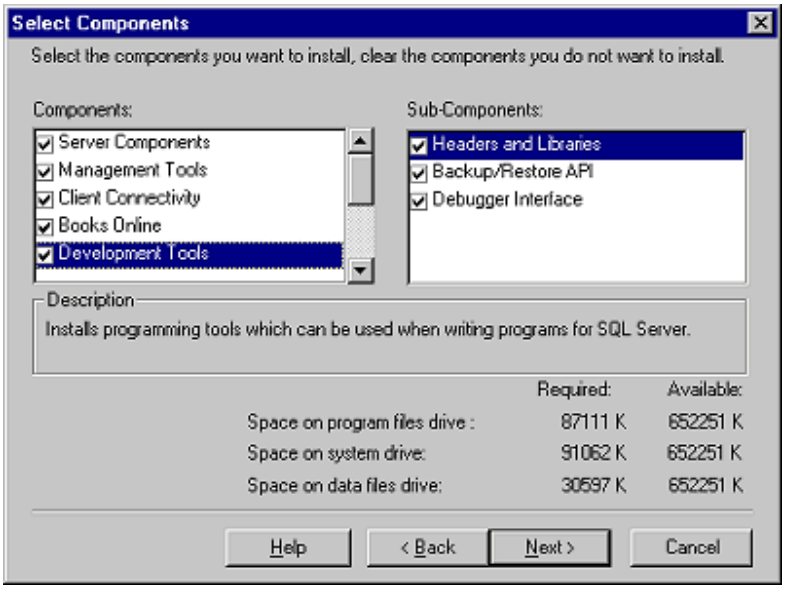


6. Depois de passar por estas telas introdutórias, aparece a primeira tela da instalação do SQL Server.

7. O programa de instalação é um assistente que te encaminha pelo processo de instalação. Embora o processo seja o mesmo que o da instalação do SQL Server, você não repetirá exatamente os mesmos passos. Aqui, você quer instalar apenas as ferramentas de gerenciamento. Clique em Next para continuar.
8. Aparece a tela do acordo de licença de software. Clique em Yes para continuar.
9. Agora, você deve digitar seu nome e o nome da sua organização, e a seguir o número de série do produto. Clique em Next para continuar.
10. Agora, aparece a tela de tipo da instalação [Setup Type].



11. Para instalar apenas as ferramentas administrativas, escolha Custom e clique em Next. Antes de prosseguir, verifique se as pastas escolhidas estão corretas, e se você tem espaço em disco suficiente.Se não, mude as pastas ou discos, clicando no botão Browse.
12. Na tela de seleção de componentes, você deve selecionar os componentes que você



deseja instalar e desmarcar os que você não instalará.

Geralmente, você desmarcará Server Components (afinal, você só quer as ferramentas de administração, e não o SQL Server). Selecione Client Connectivity, e Management Tools. Do lado direito da tela, você pode desmarcar componentes individuais das mesmas. Depois que você escolher o que achar adequado, clique em Next para continuar.

13. Finalmente, a instalação vai começar. Lhe é mostrada a tela final, onde você pode conferir as opções que você selecionou. Se estiver satisfeito com as seleções feitas, clique em Next para começar a instalação. Caso contrário, clique em Back, e altere o que achar necessário.

Depois que você clica em Next, os arquivos são copiados para seu computador. Quando o processo de instalação finalizar, você volta apara a primeira tela do programa de instalação. Para sair dessa tela, clique em Exit.

Quando as ferramentas de gerenciamento já estivrem instaladas, você pode testar se a instalação foi bem-sucedida. Veja, se em Iniciar | Programas | Microsoft SQL Server 7.0, as ferramentas que você selecionou tem seus atalhos. Para verificar se as ferramentas funcionam, escolha alguma e a execute, vendo se você pode se conectar a um servidor remoto. Claro, para que este teste funcione, você deve ter uma conta legal e as permissões necessárias no servidor SQL Server. Se você puder fazer uma conexão com o servidor SQL Server, você confirma que as ferramentas foram instaladas com sucesso.

Na maioria dos casso, assim que as ferramentas de gerenciamento forem instaladas, você será capaz de usá-las imediatamente sem qualquer configuração adicional. Mas se a sua rede não for uma rede Microsoft pura, então você pode ter que mudar algumas opções de configuração usando a ferramenta de configuração de clientes [Client Network Utility]. Veja na seção Client Network Utility como utilizar essa ferramenta.

Registrando um servidor

Quando você tiver certeza que o SQL Server está instalado e você pode se conectar a ele (confira em verificando se você pode se logar no SQL Server), a próxima etapa é se registrar com o SQL Enterprise Manager. Enterprise Manager é o principal programa usado para gerenciar o SQL Server (veja mais sobre o SQL Enterprise Manager). Ele tem a capacidade de não apenas gerenciar um servidor SQL Server local, mas também servidores múltiplos conectados em qualquer lugar na mesm rede física. Mas antes de tornar o SQL Enterprise Manager capaz de fazer isso, você deve lhe contar sobre os diversos servidores SQL Server que você possa ter, e isso é feito através do processo de registrar-se. Quando um servidor SQL Server está registrado com uma cópia do Enterprise Manager, essa cópia do SQL Enterprise Manager é capaz de gerenciar aquele servidor SQL Server, não importando onde ele esteja na ree.

Nota: Por padrão, para administrar um servidor, você deve ser membro do grupo local Administradores no computador onde o SQL Server foi instalado. Administradores de domínio do Windows NT são membros do grupo local Administradores.

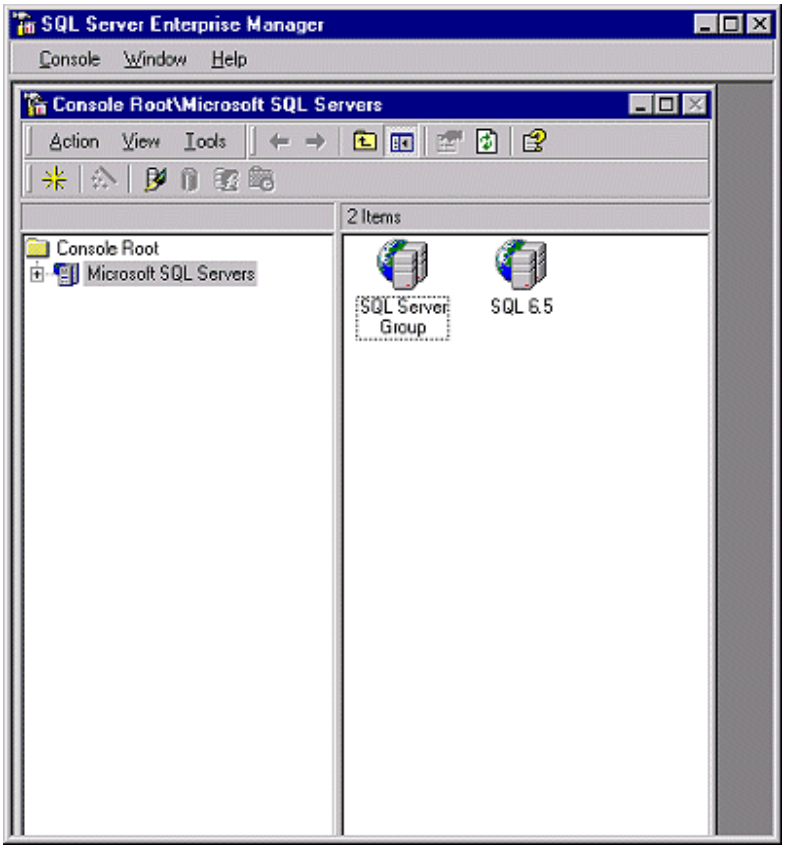
O registro é um processo que precisa ser feito apenas uma vez, embora você possa desresgistrar e registrar novamente queal servidor SQL Server quantas vezes você quiser. As informações de registro do SQL Server são mantidas no registro do Windows NT. O Enterprise Manager usa essa informação a cada vez que você se conecta com um servidor SQL Server registrado.

Há duas maneiras de registar um servidor SQL Server usando o Enterprise Manager: manualmente, ou com o Assitente de Registro [Registration Wizard]. Aqui olharemos os dois métodos, começando pelo Assistente de Registro.

Nota: Diversas cópias do Enterprise Manager podem estar distribuídas em diversos computadores ao longo da empresa. Isso pode fazer com que seja necessário você registrar diversas vezes o(s) servidor(es) que você quer administrar. Isso ocorre pois o registro é feito com o Enterprise Manager e portanto, em cada cópia do Enterprise Manager devem ser registrados os servidores que se deseja administrar.

Registrando um servidor utilizando o Registration Wizard

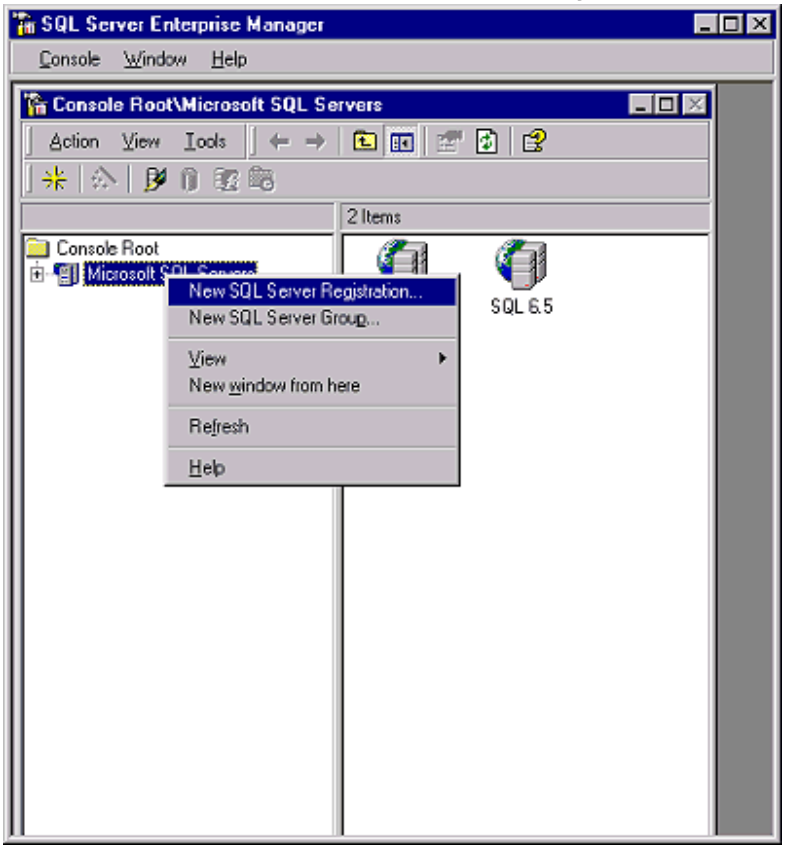
Antes de tentar resgistrar um servidor SQL Server com o SQL Enterprise Manager, confirme que o SQL Server está instalado e rodando adequadamente (ver em verificando a instalação). Se ele não estiver instalado e rodando corretamente, você não será capaz de registrá-lo com o SQL Enterprise Manager.



Registrando um servidor SQL Server com o SQL Enterprise Manager pela primeira vez.

1. A partir do Grupo de programas do Microsoft SQL Server (Iniciar | Programas | Microsoft SQL Server 7.0 | Enterprise Manager), execute o SQL Server Entreprise Manager. Isso inicia o Microsoft Management Console (MMC).

2. Agora você está pronto para iniciar o assistnte de registro [Registration Wizard]. Para iniciá-lo, clique com o botão direito em Microsoft SQL Servers, que aparece abaixo de Console Root. Aparecem então diversas opções, e você quer registrar um servidor [New SQL Server Registration...]

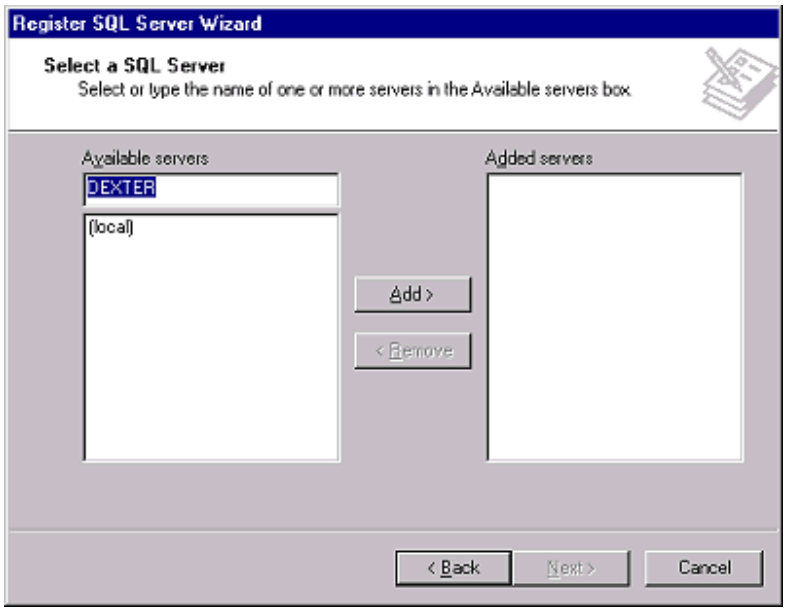


3. Depois de escolher a opção New SQL Server Registration, aparece a primeira tela do Assistente de Registro [Registration Wizard]. Clique em Next para continuar.

4. A seguir, o assistente te pde para selecionar qual servidor SQL Server que você quer registrar, como na figura abaixo:

   Os servidores SQL Server disponíveis aparecem no lado esquerdo da tela. Se o seu servidor não estiver sendo mostrado, talvez ele não tenha sido instalado adequadamente, ou os dois serviços do SQL Server podem não estar iniciados (ver em verificando a instalação se eles estão iniciados). Supondo que seu servidor esteja listado (local é o servidor SQL Server local), clique no servidor que você quer registrar e clique em Add. Isso move o servidor SQL Server para o lado direito da janela, embaixo de Servidores Adicionados [Added Servers]. Você pode registrar mais de um servidor de uma vez se você quiser.
5. Depois que você tiver adicionado os servidores SQL Server desejados, clique em Next, e vai aparecer uma janela perguntando a você qual opção de conexão que você quer utilizar para se conectar ao SQL Server.

   Você tem duas opções: autenticação do Windows NT ou autenticação do SQL Server. Se esta é a primeira instalação do SQL Server na sua organização, escolha por agora autenticação do SQL Server; isso pode ser mudado mais tarde se você mudar seu modo de segurança. Mas se não for a primeira instalação do SQL Servre, escolha o modo de segurança que os outros servidores SQL Server estiverem utilizando. O exemplo a seguir supõe que você tenha escolhido a autenticação do SQL Server [SQL Server authentication]. Depois de feita sua escolha, clique em Next.
6. A seguir, o assistente quer que você escolha se o SQL Enterprise Manager se lembre



   do seu nome de login e senha.

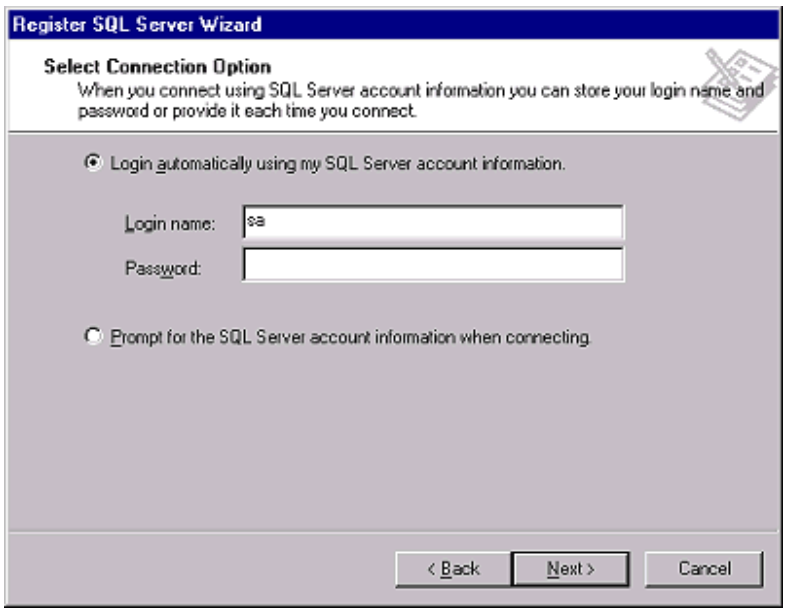
A primeira opção, "Efetuar logon automaticamente usando minha informação de conta do SQL Server" [Login Automatically Using My SQL Server Account Information], pode ser escolhida para que o Enterprise Manager se lembre de sua senha e nome de login. Assim, você não precisa de, a cada vez que iniciar o Enterprise Manager, ficar digitando essa

informação novamente.
Se você não quiser que o Enterprise Manager lembre-se de seu login e senha, escolha a opção "Perguntar a informação da conta do SQL Server quando se conectar" [Prompt for the SQL Server account information when connecting].
Se você escolher a primeira opção, digite SA como nome de login, e deixe a senha [Password] em branco. Você deve usar esse nome de login e senha quando registra pela primeira vez um novo servidor SQL Server poque você ainda não atribuiu à conta SA uma senha, nem criou qualquer outro nome de login. Clique em Next para continuar.

7. Você deve adicionar um servidor SQL Server a um grupo de servidores.

Nessa tela, você pode adicionar ao grupo padrão SQL Server Group, ou a um outro grupo preexistente (selecionando a primeira opção). Você pode ainda criar um novo grupo (selecionando a segunda opção [Create a new top-level SQL Server group to add the SQL Server(s) to.]), e seu servidor será adicionado a esse grupo. Grupos de servidores são usados para agrupar servidores SQL Server de usos semelhantes, para fins de aministração, e são completamente opcionais. Tudo que os grupos de servidores fazem é agrupar grupos de servidores semelhantes para visualização no Enterprise Manager. Escolha a opção mais adequada e clique em Next para continuar.

8. O assitente de registro mostra sua última tela.



Se você quiser fazer quaisquer mudanças, você pode fazê-las clicando no botão Back. Ou, se tudo estiver conforme você queria, clique em Finish para concluir. A caixa de diálogo de registro do SQL Server aparece, e você recebe uma mensagem dizendo se o registro foi bem-sucedido. Se você receber uma mensagm de erro, veja a seção de solução de problemas.

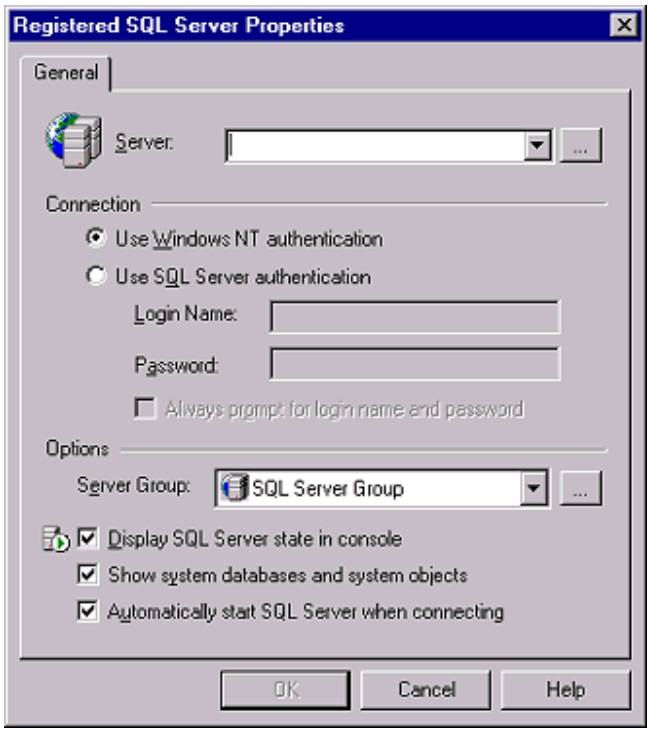
9. Clique em Close, e volta-se para o SQL Entrprise Manager. Embaixo de Console Root, você verá o cabeçalho Microsoft SQL Servers. Clique no sinal de mais perto deste cabeçalho, e um ou mais grupos de servidores serão mostrados. O(s) servidor(es) que você acabou de registrar aparecerão dentro do grupo que você definiu para ele(s). Supondo que você não tenha criado nenhum grupo e tenha adicionado-os ao grupo padrão, o único grupo listado será o padrão SQL Server Group. Para ver o servidor que você acabou de registrar, clique no sinal de mais perto do grupo de servidores em que você adicionou o(s) servidor(es). Se o servidor não aparecer dentro de nenhum grupo de servidores, ele não foi registrado corretamente com o Enterprise Manager.

Registrando um servidor manualmente

Se você não quiser usar o Assistente de Registro, você não precisa.
Registrando um servidor manualmente com o Enterprise Manager

1. O primeiro passo é desligar o Assistente de registro. Depois que ele estiver desligado, então dessa vez em diante, você será capaz de registrar manualmente qualquer servidor com o Enterprise Manager. Para desligar o assistente de registro, você deve primeiro iniciá-lo clicando com o botão direito no texto Microsoft SQL Server, ou em qualquer grupo de servidores, ou em qualquer servidor registrado, e então selecionar a opção Novo registro de servidor [New SQL Server Registration]. Isso mostra a primeira tela do Assistente de registro.

2. Para desligá-lo, selecione a opção "A partir de agora eu quero realizar essa tarefa sem usar um assistente" [From now on I want to perform this task without using a wizard]. Clique em Next e na outra janela em Cancel. Isso fechará o assistente de registro e também o impedirá de ser executado a cada vez que você for registrar um servidor.
3. Agora, para regsitrar um servidor manualmente, clique com o botão direito no texto Microsoft SQL Server, ou em qualquer grupo de servidores, ou em qualquer servidor registrado, e então selecionar a opção Novo registro de servidor [New SQL Server Registration]. Isso mostra a caixa de diálogo "Propriedades do servidor SQL registrado" [Registered SQL Server Properties]



4. Complete essa janela usando o mesmo tipo de informações que foram descritas quando descrevemos o Assistente de registro. Uma diferença entre o Assistente de registro e essa caixa de diálogo são as três opções na parte de baixo da janela.
   Display SQL Server State in console - Esta opção, se selecionada, faz com que o Entrprise Manager regularmente interrogue o serviço MSSQLServer para saber se ele está rodando, e mostra uma luz verde no ícone no console quando ele está rodando, e uma luz vermelha se o serviço não estiver sendo executado.
   Show System Databases and System Objects - Se esta opção for escolhida, todos os bancos de dados e objetos do sistema serão mostrados no console. Caso não seja escolhida, os mesmos não aparecem no console.
   Automaically Start SQL Server when connecting - Assumindo que o serviço MSSQLServer não esteja ajustado para iniciar automaticamente, esta opção pode ser usada para iniciar automaticamente este serviço quando da primeira conexão com o servidor.
   Por padrão, as três opções estão escolhidas agora, e são escolhidas automaticamente quando se usa o assistente de registro. Você pode fazer qualquer escolha que achar adequada. Quando você tiver completado suas escolhas, você pode registrar o servidor clicando em OK.

Como editar as informações de registro do servidor SQL Server

Às vezes, você pode precisar de editar as configurações de registro do SQL Server, como quando você mudar o login ou a senha que você usou originalmente para registrar o servidor.

Fazendo mudanças em um servidor registrado

1. No Enterprise Manager, selecione o servidor cujo registro você quer alterar.

2. Clique com o botão direito no nome do servidor e então escolha Edit SQL Server Registration no menu. Isso mostra a caixa de diálogo Registered SQL Server Properties, que foi mostrada na figura acima.
3. Faça quaisquer mudanças necessárias. Quando terminar, clique em OK para salvar essas configurações e voltar para o Enterprise Manager

Como cancelar o registro de um servidor

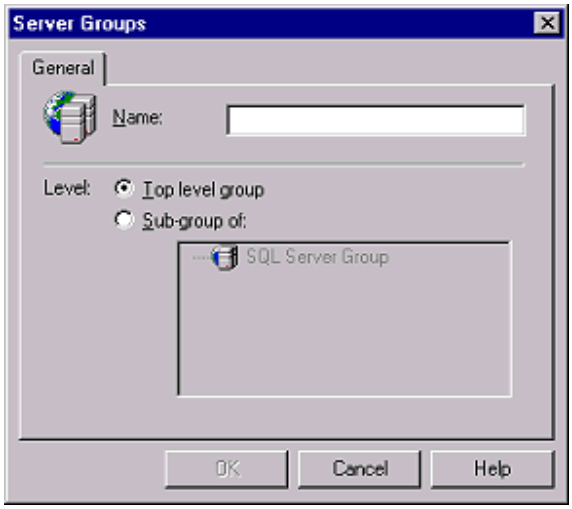
De tempos em tempos, pode ser necessário cancelar o registro de um servidor, no Enterprise Manager. Para isso, faça:

1. No Enterprise Manager, selecione o nome do servidor cujo registro você quer cancelado.
2. Clique com o botão direito em seu nome, e selecione Delete do menu.
3. Uma caixa de confirmação aparece, pedindo-lhe para clicar em Yes para remover o servidor, ou No para cancelar a operação. Clique em Yes para cancelar o registro do servidor.

Gerenciando grupos de servidores

Embora você possa criar novos grupos a partir do assistente de registro, ou a partir da caixa de diálogo Registered SQL Server Properties, você também pode crirar, renomear e excluir grupos de servidores manualmente. Você também pode mudar um servidor de um grupo para outro se você quiser. Para gerenciar os grupos de servidores, faça o seguinte:

1. A partir do Enterprise Manager, clique no sinal de mais perto do cabeçalho Microsoft



SQL Servers. Isso mostra todos os grupos de servidores atualmente embaixo dele. Clique com o botão direito no nome Microsoft SQL Server, ou em qualquer grupo de servidores, e então selecione Novo grupo de servidores [New SQL Server Group] do menu. Aparece a caixa de diálogo de Grupos de Servidores.

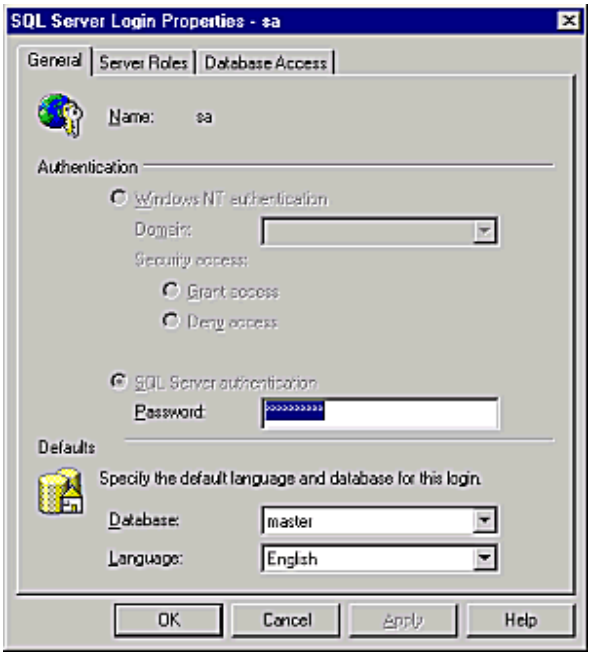
2. Para criar um novo grupo de servidores, entre com o nome do novo grupo na caixa Name e clique em OK. Isso fará esse grupo automaticamente um grupo de servidores do nível mais alto. Se você quiser, você pode escolher criar um grupo como subgrupo de algum outro grupo, escolhendo a opção "Sub-grupo de" [Sub-group of:]. Na maioria dos casos, grupos de servidores do mais alto nível são mais do que suficientes.

3. Se você quiser remover um grupo de servidores, clique com o botão direito no nome do grupo de servidores que você quer remover, e então selecione Delete do menu. O grupo é removido imediatamente sem nenhum aviso.

Atribuindo uma senha ao Administrador do Sistema (SA)

Se você estiver usando o método de autenticação do NT, você não precisa executar este passo porque não se exige que você efetue logon no servidor SQL Server diretamente usando um nome de login [login ID].
Mas se você estiver usando o método de autenticação do SQL Server, uma das primeiras coisas que você vai querer fazer é definir uma senha para a conta SA. O SA tem a capacidade de executar qualquer função no SQL Server, e você deve evitar que usuários não autorizados efetuem logon no servidor SQL Server.
Colocando uma senha para a conta SA

1. Do Enterprise Manager, abra a pasta que representa o servidor SQL Server, cuja senha do SA precisa ser alterada.
2. Abra a pasta Security [Segurança], e clique uma vez em Logins, e então no lado direito da janela, os logins atualmente disponíveis são exibidos.



3. Clique com o botão direito no nome de login SA, e selecione Propriedades [Properties] no menu; a caixa de diálogo Propriedades aparece.

4. Para adicionar uma senha paa a conta SA, digite-a na caixa identificada por Password. Você vai notar que a caixa da senha está preenchida com asteriscos. Isso não significa nada, já que atualmente não há senha para o login SA. Escolha uma senha que não seja fácil de adivinhar.
5. Depois que você tiver colocado uma senha, clique em OK. Isso salvará a senha e fechará a caixa de diálogo. Agora você precisa de voltar ao registro desse servidor e editá-lo para indicar a nova senha da conta SA. (para editar o registro do servidor, faça como indicado em Registrando um servidor manualmente)

O restante desta tela de Propriedades será visto no decorrer do curso.

Agora, o SQL Server está funcionando e pronto para ser configurado para qualquer aplicação na qual ele esteja sendo utilizado.

Solução de problemas de instalação

Se você seguiu corretamente os avisos e instruções até aqui, você não deve encontrar problemas na instalação do SQL Server. Quase todos os prbblemas que você encontrar são porque você deixou passar uma etapa, ou cometeu algum engano durante o processo de instalação. Aqui, vamos comentar alguns prblemas comuns de instalação e como corrigí-los.

Como identificar problemas na instalação do SQL Server

Há muitas maneiras de identificar que sua instalação do SQL Server falhou. Entre eles:

- Mensagens de erro. A resposta mais comum que o SQL Server te fornece são mensagens de erro. Embora as mensagens que você receba possam não ser sempre precisas, a primeira suposição que se deve fazer quando receber uma mensagem de erro é que ela é precisa e que você precisa encontrar a causa dela. Algumas mensagens são óbvias, já outras são bem obscuras. Se você não puder determinar a partir da própria mensagem, qual é o problema, escreva a mensagem por inteiro e procure nos livros on-line do SQL Server (SQL Server books on-line). Se lá você não encontrar a mensagem, tente olhar na Microsoft TechNet (você tem que ser um assinante para recebê-la) ou no site da Microsoft na Internet.
  Se você não encontrar a mensagem de erro, você pode checar o Visualizador de Eventos [Event Viewer] do Windows NT, no log de aplicativo, para alguma mensagem relacionada. Você ainda pode ver os logs de erro do SQL Server, se houver algum, para pistas. Os logs de erro do SQL Server estão localizados na pasta \log abaixo da pasta onde o SQL Server foi instalado. Encontre o arquivo chamado Errorlog e abra-o com o Notepad ou Wordpad.
  Há também o arquivo Sqlstp.log, nas pasta C:\WINNT, que lhe dá informações sobre o processo de instalação do SQL Server.
  Todos os logs de erros são arquivos ASCII e podem ser facilmente visualizados com qualquer editor de textos, Os logs de erros do SQL Server são difícieis de interpretar, mas podem te dar uma dica do que aconteceu.
- Não foi possível verificar a instalação. Se você tentou verificar a instalação como descrito em Verificando a instalação, e o problema parece ser algo faltando da instalação que deveria estar aí, você tem duas opções. Ou execute o programa de instalação novamente sobre a instalação atual, ou você pode primeiro excluir a instalação danificada, e então reinstalar.
- Verificar o arquivo Cnfgsvr.out. Esse arquivo, localizado na pasta \INSTALL, abaixo da pasta onde o SQL Server foi instalado, é um arquivo de saída gerado pelos scripts que rodam durante a instalação e grava mensaegns de erro DBCC.

A seguir são descritos alguns dos problemas mais comuns que são encontrados quando da instalação do SQL Server.

Uso da versão errada do NT Server

O uso da versão errada do NT Server, ou o uso de um Service Pack antigo, pode causar uma grande variedade de prblemas, muito deles difíceis de diagnosticar. Se a instalação produz mensagens de erro obscuras sem razão aparente, assegure-se de que a versão do NT Server

que você está usando seja adequada. Se você descobrir que tem a versão errada, desinstale o SQL Server, atualize o NT Server para uma versão aceitável, e então reinstale o SQL Server.

Arquivos abertos durante a instalação

Durante o processo de instalação, o SQL Server substitui alguns arquivos do NT Server. Se algum desses arquivos estiver aberto durante a instalação, eles podem causar a exibição de uma mensagem de erro crítico. Essa é a razão de ser importante cetificar-se de que não há nenhum outro programa rodando quando o SQL Server for instalado. Se você descobrir que tem um ou mais utilitários do NT aberto que pode estar causando o erro, feche-os, e clique no botão de Retry mostrado pela mensagem de erro. Se isso não funcionar, você pode ter que abortar a instalação do SQL Server e tentar de novo, desta vez sem nenhum programa rodando.

Os serviços MSSQLServer ou SQLServerAgent não iniciam

Este é provavelmente o problema mais comum encontrado quando se instala o SQL Server. Siga os passos abaixo para te ajudar a determinar a possível causa desse problema:

- Você criou uma conta de serviço como descrito na instalação do software de servidor?
- A conta de serviço foi criada adequadamente, com direitos administrativos e os outros direitos avançados exigidos?
- A conta de serviço foi criada no domínio de contas correto do NT?
- A conta de serviço foi informada corretamente quando pedida durante o processo de instalação? Você usou acidentalmente sua conta de logon on NT como a conta de serviço?
- Você digitou tudo corretamente?

Se você não conseguir descobrir o problema, delete a conta de serviço que você criou e crie uma nova, seguindo cuidadosamente as recomendações da instalação do software de servidor. Então vá para o Painel de Controle, onde estão os serviços, e asegure-se de que você selecionou a conta de servço para os dois serviços, junto com as senhas corretas. Com frequência, esse é um erro simples que é facilmente corrigido.

3 - Ferramentas de gerenciamento do SQL Server

MMC - Microsoft Management Console

Enterprise Manager

Service Manager

Client Network Utility

Server Network Utility

Performance Monitor

Pofiler

Query Analyzer

Books online

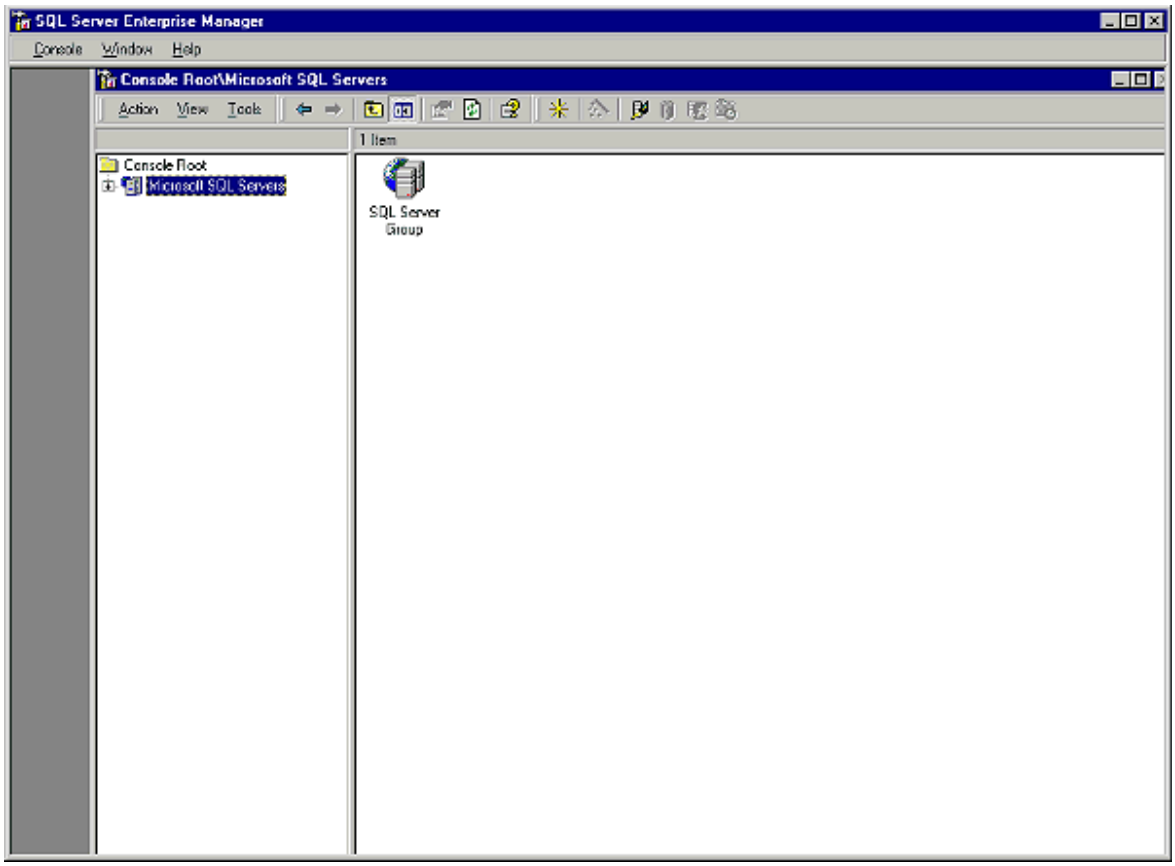
Assistentes do SQL Server

Objetivos:

- Conhecer os conceitos de alocação de espaço usados pelo SQL Server;
- Ter uma visão geral dos itens que compõem o catálogo do sistema;
- Saber o que é um banco de dados e o que ele contém;
- Aprender a criar, usar e gerenciar dispositivos de banco de dados.

SQL Server Enterprise Manager

O "SQL Server Enterprise Manager" é a porta de entrada para a Interface de usuário do SQL Server. Para iniciá-lo, selecione Iniciar, Programas, Microsoft SQL Server 7.0, Enterprise Manager. Aparece o Enterprise Manager dentro do MMC, como abaixo.



Aí temos uma porção de menus e botões. Os itens de menu importantes são Action, View e Tools:

- Action te permite fazer coisas tais como registrar um novo servidor ou um novo grupo (conforme visto na seção de instalação).
- Views te fornece uma lista dos diferentes tipos de visões disponíveis para você. Você pode selecionar as visões grande, pequeno, detalhe ou lista dos ícones e suas propriedades associadas. Ainda é possível definir quais itens e quais barras de ferramentas você verá. Exatamente como no Windows Explorer.
- O menu Tools lista todas as ferramentas e assistentes do SQL Server. Você pode fazer backup de um banco de dados; parar, iniciar e configurar a replicação; e iniciar ferramentas como o Query Analyzer (Analisador de consultas), entre outras.

Imediatamente à direita dos menus há outros itens de barras de ferramentas. Estes são basicamente atalhos para os itens mais usados da barra de menu. Temos, entre outros:

Anterior
Próximo

Atualizar

Registrar Servidor

Novo Banco de Dados

Novo Login

Query Designer

O SQL Server 7.0 tem uma ferramenta muito útil, que se parece com o Query By Example (QBE) do Microsoft Access, e é um ótimo substituto para a MS Query (do SQL Server 6.5). O nome dessa aplicação é Query Designer e faz parte das Ferramentas Visuais de Banco de Dados. É uma ótima ferramenta, mas não está listada como uma ferramenta do SQL Server, e é um pouco difícil achá-la diretamente. Para localizá-la, faça assim:

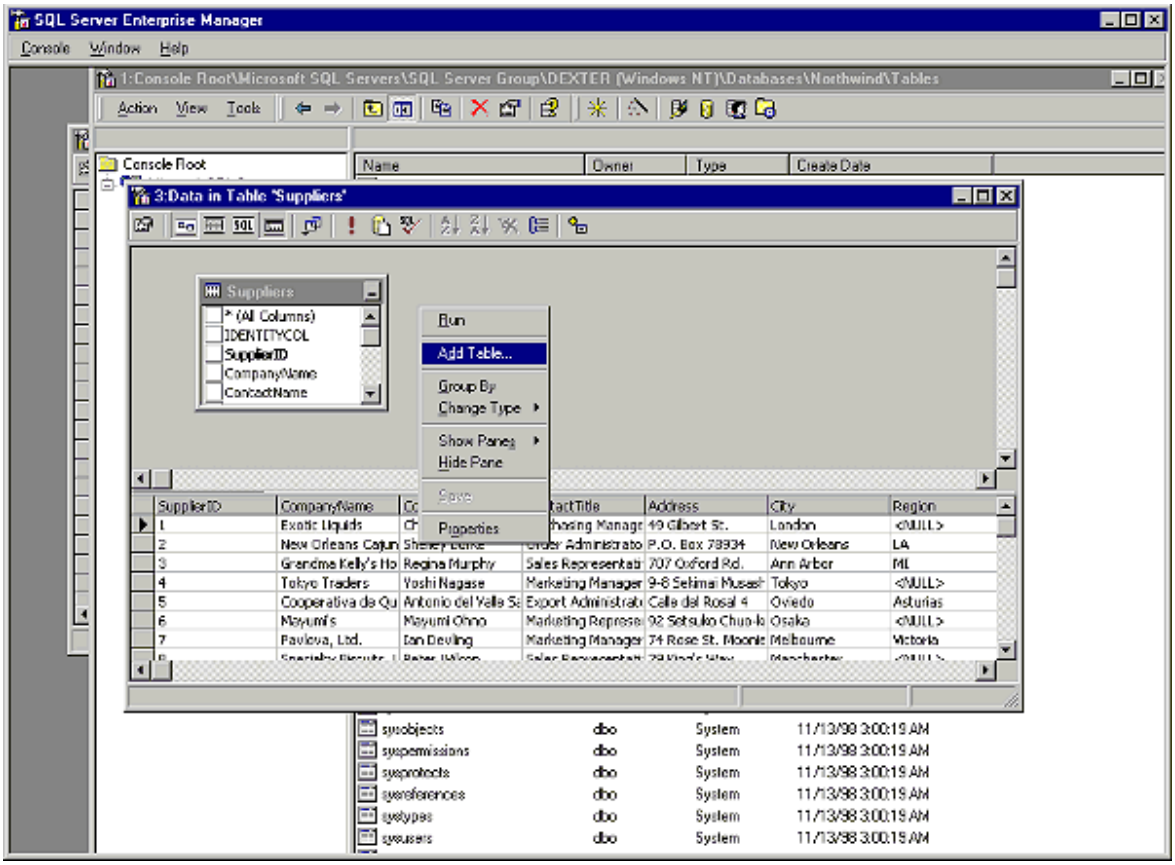
- Do Enterprise Manager, expanda o banco de dados Northwind, e expanda as tabelas.
- Clique com o botão direito na tabela Categories.
- Selecione Open Table, e então Return All Rows.

Nota: O que foi citado acima para a tabela Categories, do banco de dados Northwind, aplica-se a qualquer tabela, de qualquer banco de dados, assim como não é necessário selecionar Return All Rows. Poderia ter sido selecionado Return Top... que te perguntaria quantos registros você quer ver. A diferença é no comando SQL gerado para cada uma das seleções. Depois que você abrir a tabela, toda a janela estará envolvida com o Query Designer. Ele permite que você veja propriedades da consulta, execute consultas de seleção, de ação (inserção, atualização, exclusão, e criação de tabelas), validação da sintaxe SQL, ordenação, filtragem, e agrupamentos.

Provavelmente, estaremos vendo a seção dos Resultados (Results Pane), acionada com o botão  , que mostra os resultados atuais da consulta. Porém, ainda há outras três seções para se escolher ou adicionar à janela atual. Estas são, Show Diagram, Show Grid, e Show SQL Pane. Elas são selecionadas pelos botões no topo da janela.

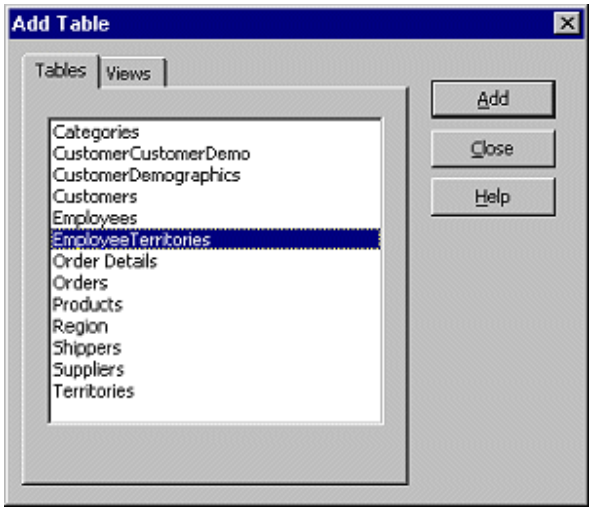
Show Diagram Pane 

Esta seção te permite visualizar graficamente o banco de dados ou as tabelas. Por padrão, só

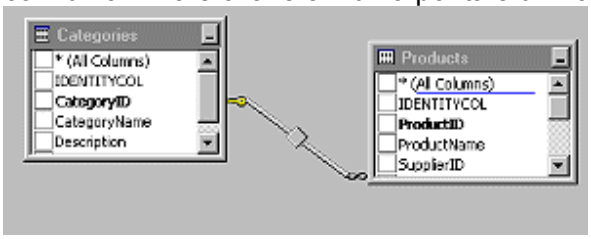


a tabela atual que estiver aberta será mostrada nesta seção. Para adicionar outr tabela, simplesmente clique com o botão direito em qualquer lugar da seçao (exceto na representação gráfica de alguma tabela). Aparece um menu com algumas opções.

Selecione Add Table. Isso mostrará todas as tabelas e visões (views) disponíveis no seu banco de dados, conforme abaixo.



Adicione a tabela Products (supondo que você seguiu o exemplo acima e está vendo as tabelas do banco de dados Northwind), selecionando-a e clicando em Add, ou dando um duplo clique no nome da tabela . Note que o relacionamento também é representado graficamente com uma linha e chave em uma ponta e um ícone de infinito na outra ponta.



Isso mostra um relacionamento de um para muitos(1:N) com CategoryID na tabela Categories como a chave primária. Pode-se selecionar para a consulta, colunas individuais ou todas as colunas marcando as caixas de verificação apropriadas perto do nome de cada coluna.

Show Grid Pane 

Esta seção permite detalhar mais a consulta sendo criada. Por exemplo, podemos selecionar a ordem para uma coluna particular, seu alias (nome que será mostrado no grid). Tudo isso de uma maneira visual.

Show SQL Pane 

Esta seção mostra a declaração SQL gerada pelas seleções feitas nas seções acima. Pode-se agora copiar e colar código desta janela, para o Query Analyzer, pra um ambiente de programação, entre outros.

Nota: A seção de resultados (Results Pane) não se atualiza automaticamente. Para atualizá-la, clique no ponto de exclamação vermelho (Run) 

No Query Designer, podemos criar consultas complexas muito mais rapidamente que no Query Analyser (que será visto adiante). Depois de criarmos as consultas, podemos colocá-las no Query Analyzer e testar sua performance.

SQL Server Client Network Utility

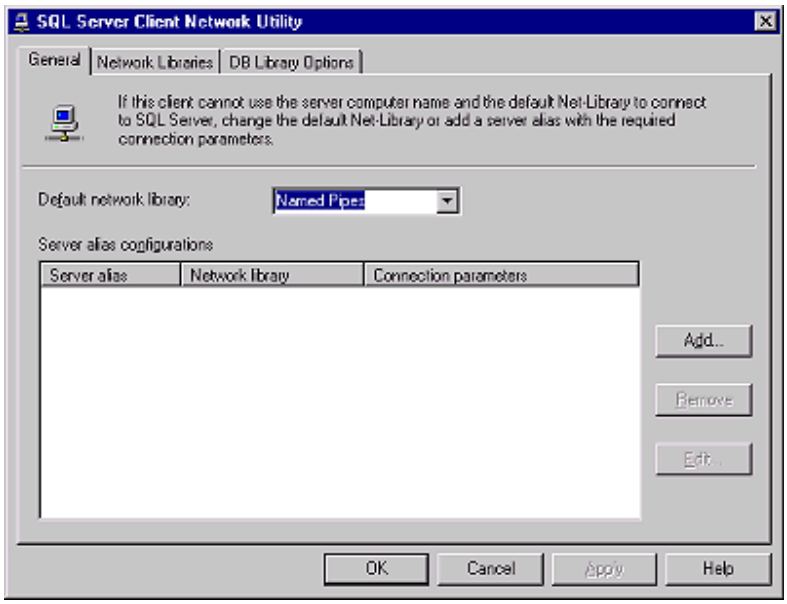
A ferramenta SQL Server Client Configuration é utilizada para configurar as ferramentas de gerenciamento, de modo que elas possam comunicar-se com sucesso com um servidor SQL Server.
A ferramenta SQL Server Client Configuration Utility está localizada no grupo de programas do SQL Server 7.0 (Iniciar | Programas | Microsoft SQL Server 7.0 | Client Network Utility. Essa ferramenta é chamada de *Client Configruation Utility* e de *Network Configuration* nos livros on-line.
Na maioria das situações, você não precisará de executar este programa porque as configurações padrão que foram utilizadas durante a instalação vão funcionar na maioria das configurações de estações de trabalho. Mas se você descobrir que as suas ferramentas de cliente não conseguem se comunicar com o servidor SQL Server, você pdoe utilizar esse utilitário para configurar adequadamente o cliente de modo que ele possa comunicar-se com o servidor SQL Server.

Iniciando a Client Network Utility

Provavelmente, esta ferramenta foi instalada quando você instalou as outras ferramentas de gerenciamento em uma estação de trabalho. Se você não instalou esta ferramenta, você deve instalá-la seguindo os passos descritos em Instalando as ferramentas de cliente. Uma vez que a Client Netowrk Utility esteja instalada, execute-a seguindo os passos abaixo:

1. Efetue lgon na estação de trabalho usando uma conta com as permissões adequadas.
2. Para executar a Client Network Utility, clique em Iniciar | Programas | Microsoft SQL Server 7.0 | Client Network Utility.

3. Aparece a janela da Client Network Utility.



4. Essa ferramenta inclui três guias que separam cada uma das três opções principais. Elas são General, Netowrk-Libraries, e DB-Library Options. A seguir descreveremos cada uma delas.

Geral

A guia geral (figura acima) tem duas seções. Na parte superior da tela você pode especificar qual Net-Library você quer usar como o protocolo padrão de rede para este cliente. A segunda parte da tela é utilizada para especificar configurações opcionais do protocolo de rede/
Como você deve se lembrar, quando o SQL Server foi instalado, você teve que especificar uma ou mais Net-Libraries para serem instaladas. Este é o software utilizado para estabelecer uma conexão de rede entre o servidor SQL Server e o software de cliente. O SQL Server usa as bibliotecas de rede [Net-Libraries] para se comunicar com um protocolo de rede específico e enviar pacotes através da rede entre um cliente e um servidor. O servidor escuta simultaneamente em diversas portas, enquanto o cliente se comunica com o servidor usando uma Net-Library específica. Para que um cliente se conecte a um servidor, ele deve usar alguma das Net-Libraries que o servidor tem instaladas.
Você pode fazer com que cliente e servidor usem a mesma Net-Library de duas maneiras: adicionar no cliente, uma Net-Library que está instalada no servidor; ou o contrário: no cliente, instalar uma Net-Library que esteja instalada no servidor. Normlamente, é mais fácil adicionar a Net-Library ao servidor, do que aos clientes.
A configuração de uma Net-Library no cliente, para se comunicar com um servidor, é opcional. Por padrão, Named Pipes é a Net-Library dos clientes, instalada durante a instalação do SQL Server para computadores executando Windows NT ou Windows 9x. Named Pipes deve funcionar bem na maioria dos casos. Mas, se esse protocolo não conectar-se com seu servidor, você vai precisar de reconfigurar o cliente com a configuração correta.
Então, caso você precise mudar o protocolo de rede para uma estação de trabalho (cliente), você pode fazer isso selecionando o protocolo apropriado da lista Protocolo de rede padrão [Default network Protocol].

A Net-Library de cliente padrão para os clientes SQL Server fazendo conexões remotas é Named Pipes, a qual não é suportada em servidores rodando Windows 9x. Clientes conectando-se com servidores rodando Windows 9x devem usar a ferramenta SQL Server Client Network Utility para executar uma das seguintes opções:

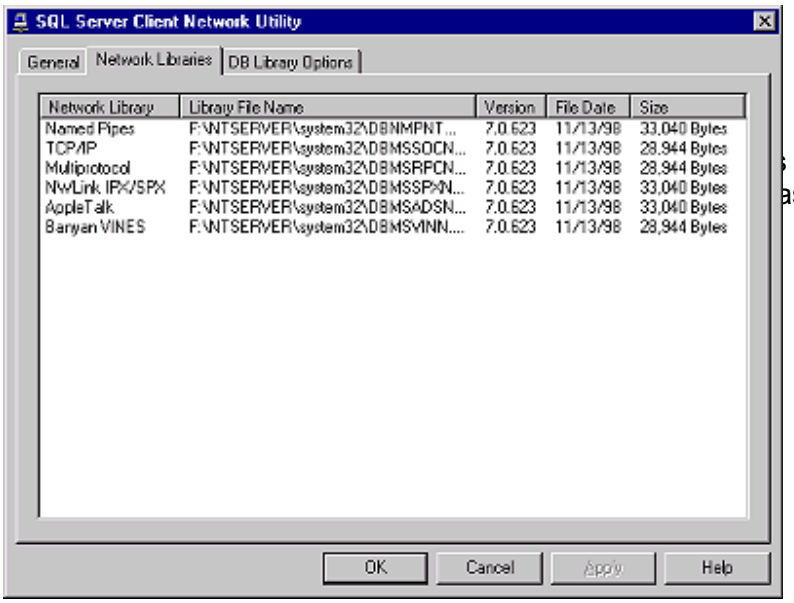
- Mudar a Net-Library padrão do cliente.
- Definir uma entrada de configuração para uma Net-Library de cliente na qual o servidor rodando Windows 9x esteja escutando.

A parte de configuração do protocolo de rede da guia Geral [General] só é usada em casos especiais. Você só vai utilizá-la se se ver em uma das seguintes situações:

- As ferramentas de gerenciamento esrão em uma estação de trabalho rodando Windows NT, conectando-se a um servidor executando Windows 95.
- Você precisa adicionar uma configuração exclusiva de protocolo para as comunicações entre um servidor SQL Server específico e um cliente executando as ferramentas de gerenciamento.
- O servidor SQL Server com o qual você quer se comunicar a partir de um cliente escuta em uma oprta não-padrão.

Normalmente, você não usará esta opção.

Nota: Para o processo Servidor, o SQL Server escuta as Net-Libraries Named Pipes, sockets TCP/IP, e Mutliprotocolo em computadores rodando Windows NT. Entretanto, Named Pipes não é aceito em commputadores rodando Windows 9x. O SQL Server instalado em computadores rodando Windows 9x escuta as Net-Libraries sockets TCP/IP e Mutliprotocolo do servidor. Se a conexão é local com o servidor (tal como um cliente e servidor na mesma máquina), o SQL Server vai escutar então a Net-Library de Memória Compartilhada do servidor.



s Network Libraries estão
as versões. Essa guia se parece

DB-Library Options

O principal objetivo da guia DB-Librarey é lhe permitir determinar se você tem ou não a versão mais atual dos arquivos de DB-Library instalados no cliente.
Outra parte da guia DB-Library Options é formada pelas duas caixas de verificação que podem ser usadas para configurar como a DB-Library se comunica com o SQL Server. Eis o que elas fazem:

- Automatic ANSI to OEM: Quando esta opção estiver selecionada, a DB-Library converte caracateres do formato OEM para ANSI quando ocorre a comunicação do cliente para o servidor SQL Server, e converte caracteres do formato ANSI para OEM quando se comunica do servidor para o cliente. Esta opção é exigida frequentemente poruqe o conjunto de caracteres utilizado pelo SQL Server é diferente daquele utilizado pelo sistema operacional do cliente. Esta opção faz automaticamente a tradução adequada entre os dois conjuntos de caracteres.
  Essa opção deve estar selecionada se o cliente estiver executando Windows NT ou Windows 9x. Se estiver sendo utilizado um cliente Windows 3.x, esta opção não deve ser selecionada.
- Use International Settings: Quando esta opção estiver selecionada, permite-se que a DB-Library pegue as configurações de formato de data, hora e moeda do sistema operacional local ao invés de utilizar a configuração definida no código do SQL Server. Esta opção deve ser selecionada se o cliente estiver sendo executado no Windows 9x ou NT.

SQL Server Profiler

O SQL Server Profiler é uma ótima ferramenta para se ver um registro contínuo da atividade do servidor em tempo real. O Profiler monitora os eventos produzidos através do SQL Server, filtra esses eventos baseados em critérios específicos do usuário, e mostra a saída traçada na tela, em um arquivo ou uma tabela.. Você pode até repetir traçados capturados anteriormente.

Monitorando com o SQL Server Profiler

O SQL Server Profiler é uma ferramenta gráfica que permite aos adminsitradores do sistema monitor eventos de mecanismo do SQL Server. Eventos são a nova maneira de se comunicar com e do SQL Server. Com eventos de mecanismo, um objeto COM pode interceptar esses eventos e agir de acordo. Exemplos de eventos de mecanismo incluem:

- Comandos Transact-SQL: SELECT, INSERT, UPDATE e DELETE.
- Conexão, falha ou desconexão de login.
- O começo ou fim de um procedimento armazenado (stored procedure).
- O começo ou fim de um lote de comandos SQL.
- Um erro escrito no log de erros do SQL Server.
- Um bloqueio adquirido ou liberado em um objeto de banco de dados.
- Um cursor que for aberto

Os dados gerados sobre cada evento podem ser capturados e salvos em um arquivo ou uma tabela do SQL Server para análise posteriro. Para coletar os dados dos eventos de mecanismo você define traços. Exemplos de dados capturados com um traço incluem:

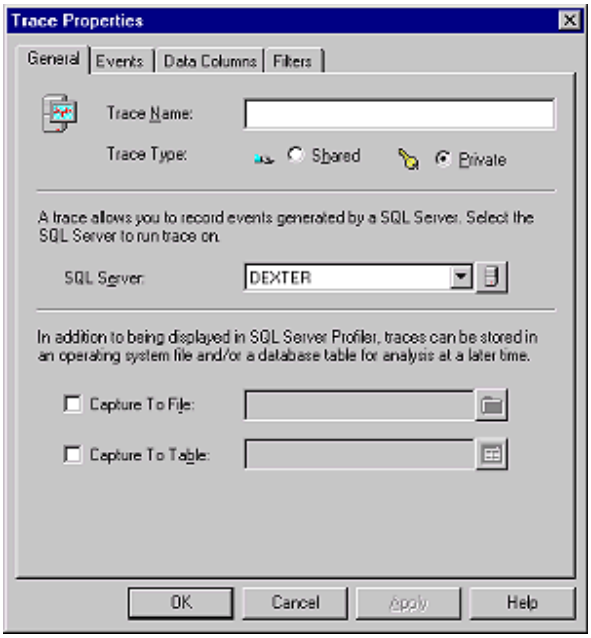
- O tipo (classe) de um evento, tal como Object:Created, o qual indica que um objeto de banco de dados foi criado.

- O nome do computador em que o cliente está rodando.
- O ID do objeto afetado pelo evento, tal como um nome de tabela.
- O nome no SQL Server, do usuário que executou ó comando.
- O texto do comando Transact-SQL ou procedimento armazenado.
- A hora em que o evento começou e terminou.

Você pode filtrar os dados de modo que apenas um subconjunto dos dados do evento seja coletado. Isso te permite coletar apenas os dados de evento em que você está interessado. Por exemplo, se você apenas está interessado em um usuário particular ou nos efeitos de um certo banco de daos, você pode filtrar esses objetos específicos e ignorar os outros. Você também pode definir filtros em itens que demorem mais do que o esperado, tal como uma consulta que demore mais do que 40 segundos.
O SQL Server Profiler também permite que os dados de eventos (capturados) sejam recolocados no SQL Server, Isso vai efetivamente reexecutar os eventos salvos como eles originalmente ocorreram.
O SQL Server Profiler pode ser usado para:

- Monitorar o desempenho do SQL Server.
- Depurar comandos Transact-SQL e procedimentos armazenados.
- Identificar consultas que executam com lentidão.
- Corrigir problemas no SQL Server. Por exemplo, você pode capturar os eventos que possam estar levando a um problema potencial e então replicar o processo em um sistema de teste para isolar e corrigir o problema.

Abaixo você vê a cauxa de diálogo inicial para a criaçao de um traço. Veremos a criação de traços em mais detalhes posteriormente.



SQL Server Query Analyzer

O Analizador de Consultas [Query Analyzer] fornece uma interface gráfica para analisar o plano de execução de uma ou múltiplas consultas, ver os dados de resultado, e recomendar indíces .
O Query Analyzer se parece com a janela Query que havia no Enterprise Manager do SQL Server 6.5. Se você já for familiarizado com a versão 6.5, essa ferramenta será bem fácil de se

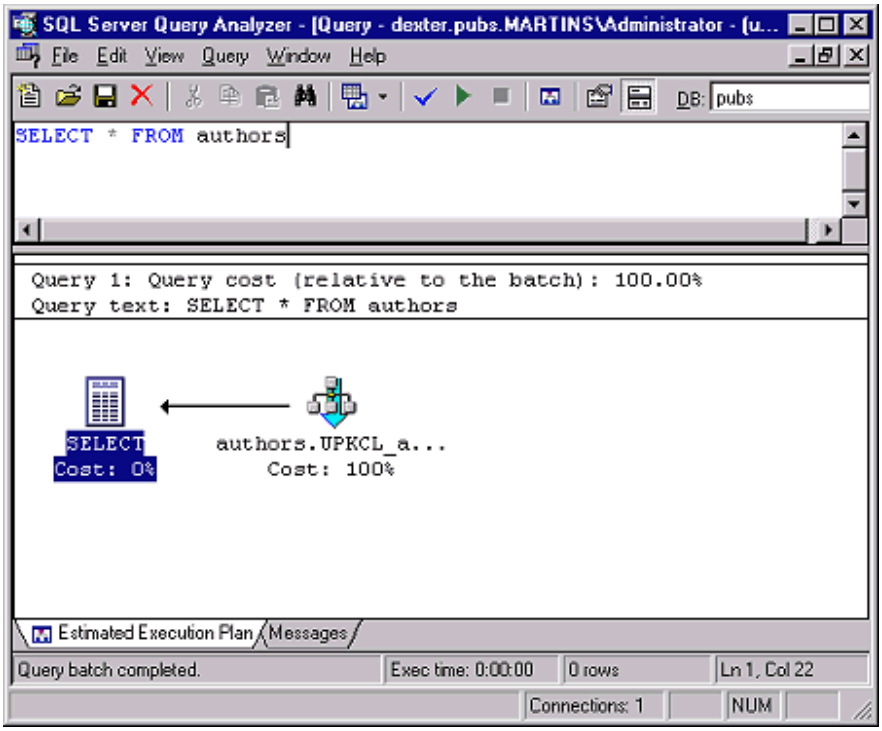
usar; mas, de qualquer maneira, aqui será explicado como utilizá-la. Essa é uma ferramenta bem fácil de se usar. Vamos começar pela execução de uma consulta simples.

Usando o Query Analyzer

1. A partir do MMC, selecione Tools | Query Analyzer. Você também pode executar o Query Analyzer pelo grupo de programas do SQL Server (Iniciar | Programas | Microsoft SQL Server 7.0 | Query Analyzer).
2. Conecte-se ao servidor SQL Server local. Efetue logon com a conta e senha (se houver) do SA que você especificou na instalação. Caso você tenha privilégios administrativos nessa máquina, você também pode se logar com a conta do NT [use Windows NT autenthication]. Dessa maneira, você também efetuará logon como SA.
3. Selecione pubs da lista na janela de consulta. Esse será o banco de dados no qual executaremos a consulta.
   Você também poderia ao invés de especificar o banco de dados, usar a declaração Use antes de sua consulta.
4. Escreva o seguinte na parte superior da janela se a janela tiver mais de uma parte.
   ```
   SELECT * FROM authors
   ```
5. Selecione o método de execução. São disponíveis quatro opções diferentes de execução:
   Execução padrão (Ctrl+T) mostra os resultados na forma de um arquivo texto, como na figura abaixo. F5 (run) executa a consulta, qualquer que seja o modo de execução selecionado.
   Executar para uma grade (Ctrl+D) fornece um formato mais agradável de se visualizar, em uma planilha, uma estrutura parecida com tabelas (um layout de linha-coluna). A figura abaixo mostra o resultado em uma grade.

   Plano de execução (Ctrl+L), mostrado na próxima figra, mostra o processo real de execução que ocorreu. Essa consulta não mostrou nada de interessante porque só havia uma tabela envolvida, mas se fosse executada uma consulta mais complexa com junções internas ou externas, você visualizaria sua representação gráifca.

Análise de Índice [Index Analysis] provavelmente lhe avisará que ele foi incapaz de

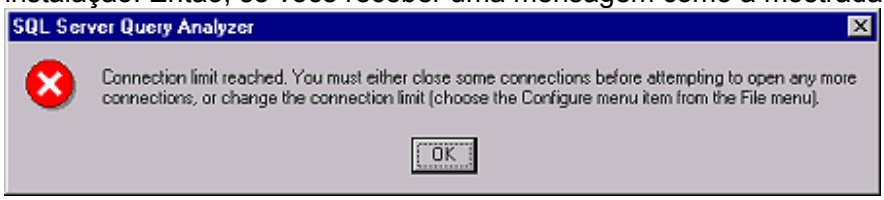


recomendar quaisquer índices. Entretanto, se você começar a escrever consultas complexas que são usadas frequentemente, você pode colá-las nesse utilitário e testá-las para sugestões de índice.

Quando você decide fazer uma nova consulta, você pode fazê-la em outra janela, clicando no botão New Quey (Ctrl+N), ou então apagar a consulta já feita e digitar novos comandos.

Note que a cada nova janela  aberta, ou seja, a cada consulta feita em outra janela, está sendo estabelecida uma nova conexão. Observe na parte inferior da janela do Query Analyzer (ao lado de Connections), quantas são as conexões estabelecidas.

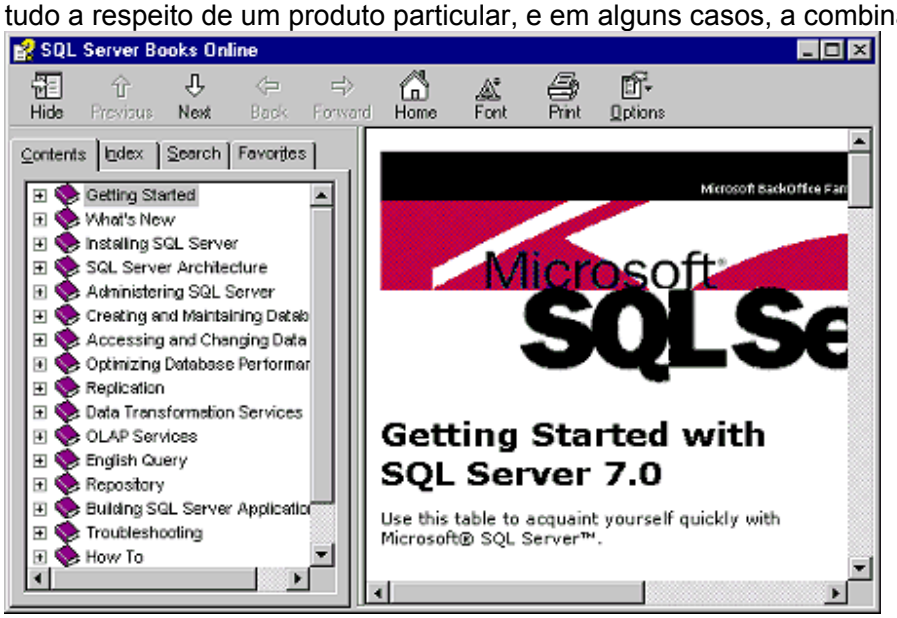
Durante a instalação, se você optou por licenciar o SQL Server como Per Server, seu servidor só suportará tantas conexões simultâneas quantas houverem sido definidas nessa fase da instalação. Então, se você receber uma mensagem como a mostrada abaixo



você deve fechar algumas conexões (janelas de consulta) antes de tentar abrir alguma nova, ou então, no menu File | Configure, deve definir o número máximo de conexões aceitas, na opção número máximo de conexões (Maximum number of connections).

SQL Server Books Online

O SQL Server Books Online é um grande recuro para ter em mãos. O guia de ajuda MS books online está se tornando o modo pardrão de acessar a ajuda com aplicações, serviços e linguagens de progrmação Microsoft. Os livros online são um lugar onde você vai encontrar tudo a respeito de um produto particular, e em alguns casos, a combinação de vários produtos.



O books online lembra o MMC ou o WIndows Explorer. O lado esquerdo tem uma visão de estrutura de árvore enquanto o lado direito tem um IE (navegador Internet Explorer). As páginas mostradas no lado direito são simples páginas HTML (você pode inclusive visuzalizar seu código fonte).

Procura por conteúdo [Contents]

Uma busca por conteúdo é como procurar em um livro baseado no sumário dos capítulos. O sumário do capítulo é visível como uma estrutura de árvore. Clicar em um livro no lado esquerdo da janela causa a abertura do livro revelando páginas, ou capítulos. Clique em uma página e a página será exibida na parte direita da janela. Clicar em um capítulo (que se parece com um outro livro) vai abrir mais capítulos, e mais páginas do lado esquerdo.

Procura por índice [Index]

A procura pelo índice vai pesquisar todas as palavras que foram indexadas quando da criação do material de ajuda. Ocasionalmente, você pode não encontrar o item que você está procurarndo e precisará de uma procura mais geral. Use a guia Pesquisar [Search] para pesquisas mais gerais.

Usando a guia Pesquisar [Search]

A guia de consulta permite que você digite uma palavra e será pesquisado em todos os documentos atrás daquela palavra. O lado esquerdo da tela vai ser preenchido com os documentos que contém a(s) palavra(s) que você pediu.

Guia Favoritos [Favorites]

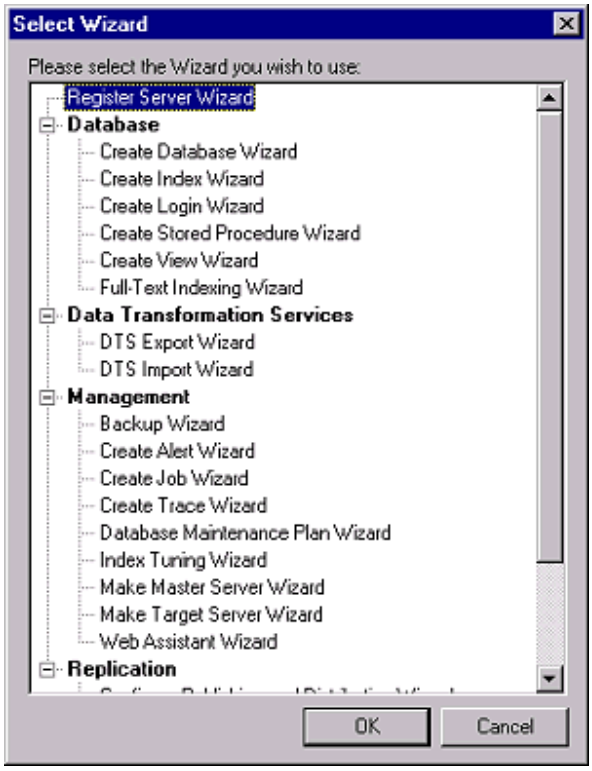
A guia favoritos é um livro de marcadores bem fácil de se usar. Se você gostou da informação que você encontrou e quer torná-las mias fácil de ser encontrada posteriormente, simplesmente selecione a guia Favoritos. A parte inferior da guia vai mostrar-lhe o tópico atual. Clique em Add para colocá-lo como parte do seu marcador. Se você precisar de excluir um item, simplesmente selecione-o e aperte o botão Remove.
Se você está inseguro a respeito de como algumas das ferramentas funcionam, ou apenas esqueceu, lembre-se de que os assistentes são ótimas ferramentas para o aprendizado. Eles fazem um trabalho excelente de apontar as tarefas necessárias que estão sendo realizadas. Depois de executar os assistents algumas vezes, você terá uma boa idéia do processo como um todo. Mesmo que você não tenha, use esta apostila ou os livros on-line.

Asistentes do SQL Server [SQL Server Wizards]

Os assistentes do SQL Server te ajudam a crirar seus objetos de banco de dados e serviços sem precisar de estar sempre olhando em suas notas sobre como fazer algo corretamente. Ter um assistente para te encaminhar em cada tarefa é uma maneira ótima de se aprender o que necessita ser feito. Quase tudo tem um assistente. Você já deve alguma vez ter evitado uma certa tarefa só porque você não tinha tempo para pesquisar como realizá-la. Como você deve saber, a não ser que você conheça o procedimento completamente, você cometerá algum erro ao realizá-lo. Seu tempo já é bem escasso sem ter que se recuperar de erros. Se você não estiver certo do que você está fazendo, ou se você está fazendo certo, deixe o assistente ser seu guia. Alguns desses assistentes são simples, enquanto outros são mais complexos. Todos os assistentes estão disponíveis no Enterprise Manager. Para ver uma lista dos assistentes

disponíveis, selecione algum servidor, e no menu Ferramentas [Tools] do Enterprise Manager, escolha a opção Assistentes [Wizards]. Aparece a janela mostrada abaixo:

Aqui serão discutidos alguns assistentes e os passos principais para o assistenete, dando o máximo de detalhe possível, mas sem exagerar. Bem, assistenstes estão aí pra te ajudar, e supõe-se que eles tornem as tarefas difíceis mais fáceis. Então, vamos começar.



Assistente de registro de servidor [Register Server Wizard]

O Assistente de registro de servidor obviamente, registra seu servidor. O ato de registrar o seu servidor é o processo de contar ao SQL Server o nome do servidor que você quer registrar, o tipo de segurança que você está utilizando, seu nome de login e senha (se não estiver usando autenticação do NT), e o grupo de servidor do qual esse servidor vai fazer parte. Você já deve ter utilizado esse assistente em Registrando um servidor.
Você pode criar um novo grupo de servidor quando da criação de um novo banco de dados. Um servidor somente pode existir como parte de um grupo.

Assistente de segurança [Security Wizard]

O assistente de segurança automatiza a tarefa de criação de logins para um servidor SQL Server. O assistente te permite especificar ou uma conta de usuário do NT ou criar uma autenticação do SQL Server. Veja mais sobre segurança.

Asistente de criação de bancos de dados [Create Database Wizard]

O assistente de criação de banco de dados automatiza a tarefa de criar um banco de dados. O assistente te acompanha nos passos principais para criação de um banco de dados. Estes

passos incluem o nome do banco de dados, os arquivos do banco de dados, os arquivos de log, o tamanho inicial dos arquivos, e como esses arquivos irão crescer. Tabelas não são criadas aqui. Elas são criadas por comandos Transact-SQL ou clicando com o botão direito em um banco de dados e escolhendo Nova Tabela [New Table]. Saiba mais sobre a criação de bancos de dados e tabelas.

Assistente de criação de alertas [Create Alerta Wizard]

Quando ocorre um erro ou quando o SQL Server dispara um evento, eles são guardados no log de eventos de aplicação, com o nome de SQL Server. O SQL Server Agent lê o log de eventos e faz uma comparação dos eventos com um alerta (que você define). Se houver uma combinação, um alerta é disparado.

Assistente de criação de trabalhos [Create Job Wizard]

Um trabalho é algo que é executado com uma freqüência regular. O trabalho pode ser um comando Transact-SQL, um executável, ou mesmo um script (VB-Script). Por exemplo, você pode querer limpar todas as visitas que o seu banco de dados monitora para uma página Web. Isso pode ser feito a cada dia, mês, semana ou ano. O trabalho pode ser simples ou bastante complexo.

Assistente de plano de manutenção de banco de dados [Database Maintenance Plan Wizard]

Este assistente creia uma série de trabalhos que ajudam o seu banco de dados a funcionar melhor. Por exemplo, ele pode agendar a realização de backups em uma base regular e checar por quaisquer inconsistências. Veja um exemplo de seu uso Agendando um backup completo de banco de dados ou de log de transações utilizando um assistente.

Assistente de criação de índices [Create Index Wizard]

Índices são criados automaticamente quando você cria restrições PRIMARY e UINIQUE; entretanto, você também pode querer criar índices em outros campos que serão consultados com frequência. Este assistente permite que você selecione uma tabela e crie índices em um campo dado. Se índices já existem, o assistente te notificará disso. Você ppode especificar o índice como um índice clusterizado (se algum ainda não existe), um índice não-clusterizado, índice único, os fatores de preenchimento também podem ser definidos. Se um item não puder ser indexado, tal como um tipo de dados imagem, que aramzaena as informações em um formato binário (que não é bom para indexação), o SQL Server bloqueará esse campo para indexação. Aprenda mais sobre índices.

Assistente de criação de procedimentos armazenados [Create Stored Procedure Wizard]

Procedimentos armazenados são consultas compiladas. Por que você as quereria e o que elas são? Toda vez que você executa uma consulta, o SQL Engine deve examinar a consulta e certificar-se de que todos os campos e tabelas são válidos. Então, ele deve se decidir a executar a consulta.
Quando você cria um procedimento armazenado, tudo isso é feito de uma vez. O fato de não ter o trabalho adicional de encontrar o melhor caminho a tomar quando da execução de uma consulta pode melhorar o desempenho. Ver mais detalhes em Procedimentos Armazenados.

Assistente de criação de visões [Create View Wizard]

Uma visão é uma tabela virtual que representa uma maneira diferente de se visualizar uma tabela. Ela pode ser usada por vários motivos, incluindo mas não limitada a mostrar apenas a informação que interessa em uma tabela muito grande. Permissões de segurança podem ser administradas para permitir que apenas os pessoal de RH veja toda a tabela que mostra salários, enquanto usuários normais enxergam apenas a informação básica na visão. Veja mais sobre visões.

Assistente de Importação/Exportação DTS [DTS Import/Export wizard]

O assistente DTS [Data Transformation Services - Serviços de Transformação de Dados] permite que você utilize facilmente o DTS para importar ou exportar informações heterogêneas utilizano OLE DB e ODBC; você também pode copiar esquemas e dados de bancos de dados entre bancos de dados relacionais.
Dado heterogêneo é o dado que é armazenado em diferentes formatos de arquivo. A utilização do DTS com OLE DB e ODBC te permite recuperar dados de um formato de arquivo e usá-los com outro formato.

Web Assistant Wizard

Este assistente gera páginas HTML baseado em dados do servidor SQL Server, consultas, procedimentos armazenado, e por aí vai. Os arquivos HTML podem ser publicados de modo que sejam visíveis na intranet da companhia ou na Internet para que o mundo veja. As páginas por si não consultam o banco de dados e assim não são dinâmicas. Entretanto, pode-se cirar trabalhos que recriem as páginas regularmente, ou um gatilho poderia lançar um atarefa para recriar a página quano um item fosse inserido, atualizado ou excluído. Veja o uso desse asistente.

4 - Fundamentos de arquitetura do SQL Server

O Catálogo do Sistema

Componentes do Banco de Dados

Estrutura dos bancos de dados

Objetivos:
- Conhecer os conceitos de alocação de espaço usados pelo SQL Server;
- Ter uma visão geral dos itens que compõem o catálogo do sistema;
- Saber o que é um banco de dados e o que ele contém.

O Catálogo do Sistema

Um *banco de dados* é uma coleção de tabelas e outros objetos relacionados. Existem dois tipos de banco de dados: os bancos de dados *do sistema* são usados pelo SQL Server para operar e gerenciar o sistema e os bancos de dados *do usuário* são usados para armazenar os seus próprios dados. O catálogo do sistema é composto de tabelas no banco de dados *master.*

Bancos de Dados do Sistema

Ao instalar o SQL Server, são criados os seguintes bancos de dados do sistema:
master Controla os bancos de dados do usuário e a operação do SQL Server. Tem como tamanho inicial 16 MB. É importante manter um backup atualizado desse banco de dados. Contém informações sobre:
- Contas de login
- Processos em execução
- Mensagens de erro
- Bancos de dados criados no servidor
- Espaço alocado para cada banco de dados
- Travas [locks] de linha ativas
- Espaço alocado para cada banco de dados
- Procedimentos armazenados do sistema

model É um modelo usado para criação de novos bancos de dados, que pode ser usado para definir padrões, como autorizações default de usuário, opções de configuração, tipos de dados etc. Sempre que um banco de dados do usuário é criado, o conteúdo de *model* é copiado para ele. Seu tamanho inicial é 2.5 MB. Esse modelo pode ser alterado.
tempdb Usado para armazenar tabelas temporárias e resultados intermediários de consultas. Geralmente o seu conteúdo é excluído sempre que um usuário se desconecta. Ele cresce automaticamente confome é necessário. Seu tamanho inicial é de 8 Mb.
msdb Usado pelo serviço SQLServerAgent, para controlar tarefas como replicação, agendamento de tarefas, backups e alertas. Contém algumas tabelas de sistema, que armazenam informações usadas pelo SQLExecutive.Seu tamanho inicial é 12 Mb.

Tabelas do Sistema

As tabelas do sistema, armazenadas no banco de dados *master* e em cada banco de dados de usuário, contêm informações sobre o SQL Server e sobre cada banco de dados de usuário. Existem 17 tabelas em cada banco de dados que formam o *catálogo do banco de dados*. Todas começam com o prefixo *sys* e contém as seguintes informações:

syscolumns	Informação sobre cada coluna de cada tabela, e cada parâmetro de procedimento.
syscomments	Para cada objeto de banco de dados (visão, regra, default, trigger, procedimento) contém o texto de sua definição.
sysconstraints	Inclui informações sobre todas as restrições usadas no banco de dados.
sysdepends	Registra as dependências entre objetos do banco de dados.
sysfilegroups	Tem uma linha para cada grupo de arquivos armazenado em um banco de dados.
sysfiles	Informações sobre cada arquivo de um banco de dados.
sysforeignkeys	Informações sobre todas as restrições de chaves estrangeiras encontradas em todas as tabelas de um banco de dados.
sysfulltextcatatalogs	Lista todos os catálogos de texto completo para esse banco de dados.
sysindexes	Informação para cada índice criado e para cada tabela sem índices, além de informações para cada tabela que possui colunas *text* ou *image*.
sysindexkeys	Informação sobreas chaves e as colunas de um índice.
sysmembers	Informações sobre os membros de cada papel.
sysobjects	Informação sobre cada objeto do banco de dados (tabelas, visões,

	procedimentos, regras, defaults e gatilhos).
syspermissions	Informação sobre permissões atribuídas a usuários, grupos e papéis em um banco de dados.
sysprotects	Permissões atribuídas à contas de segurança.
sysreferences	Informação sobre toda restrição de integridade referencial usada numa coluna ou tabela de um banco de dados.
systypes	Informação sobre cada tipo de dados (do sistema ou definido pelo usuário).
sysusers	Informação sobre cada usuário que pode ter acesso ao banco de dados.

Existem também tabelas localizadas apenas no banco de dados *master*, que compõem o *catálogo do sistema*. Elas contêm as seguintes informações:

sysallocations	Informações sobre cada unidade de alocação gerenciada pelo SQL Server
sysaltfiles	Informações sobre cada arquivo gerenciado pelo SQL Server
syscharsets	Informação sobre conjuntos de caracteres [character sets] e ordens de classificação [sort orders].
sysconfigures, syscurconfigs	Parâmetros de configuração do SQL Server.
sysdatabases	Informação sobre os bancos de dados existentes.
sysdevices	Informação sobre os dispositivos, tais como o dispositivo de fita.
syslanguages	Idiomas suportados pelo servidor.
syslockinfo	Travas (locks) ativas.
syslogins	Contas de login.
sysmessages	Mensagens de erro do sistema
sysoledbusers	Contém uma linha para cada usuário e senha mapeados em um servidor.
sysperfinfo	Informação sobre os monitores de performance.
sysprocesses	Processos em execução
sysremotelogins	Contas de login remotas.
sysservers	Servidores remotos conhecidos.

Procedimentos Armazenados do Sistema

Um *procedimento armazenado* [stored procedure] é uma seqüência de comandos da linguagem Transact-SQL, compilados e armazenados num banco de dados. Os *procedimentos armazenados do sistema* [system stored procedures] são fornecidos pelo SQL Server, armazenados no banco de dados *master* e automatizam várias tarefas comuns de gerenciamento.

Por exemplo, o procedimento *sp_databases* mostra quais os nomes de bancos de dados existentes. Para executar esse procedimento, use o Query Analyzer (Iniciar | Programas | Microsoft SQL Server 7.0 | Query Analyzer). Na página "Query" digite:

```
sp_databases
```

E clique no botão Execute. Ele mostra um resultado como:

```
DATABASE_NAME                  DATABASE_SIZE          REMARKS
------------------             --------------         --------------------
master                         17408                  (null)
model                          1024                   (null)
msdb                           8192                   (null)
pubs                           3072                   (null)
tempdb                         2048                   (null)
(5 row(s) affected)
```

Outro procedimento útil é *sp_helpdb*. Ele mostra informações sobre um banco de dados.

Sintaxe:

sp_helpdb [Banco de Dados]

Exemplo:

```
sp_helpdb master
```

O resultado será algo como:

```
name          db_size         owner     dbid     created      status
---------     ----------      -------   ------   -------      ------------------
master        17.00 MB        sa        1        Apr 3 1996   trunc. log on chkpt.


                                                                size  maxsize    growth  usage
name     fileid filename                                 filegroup ----- -------- ------- -------
-------- ------ ------------------------------------- --------- -     ----     ------  ---
-----    --     -------------------------              -----     9024  Unlimited __      data
master   1       F:\MSSQL7\data\master.mdf             PRIMARY   KB    d        10%     only
mastlog  2      F:\MSSQL7\data\mastlog.ldf             NULL      1280  Unlimited 10%     log
                                                                KB    d                  only
```

Note que todos os procedimentos armazenados do sistema têm nomes que começam com 'sp_' (abrev. de system procedure).Ao executar um procedimento que inicie com 'sp_' o procedimento será procurado no banco de dados atual, se não for encontrado ele será procurado no banco de dados Master.

Quando um procedimento inicia com 'xp_' ele é um procedimento estendido, quer dizer que não foi escrito em SQL, mas foi compilado como parte de uma DLL.

Por exemplo, o procedimento xp_cmdshell executa um programa. Para executar o procedimento é necessário que esteja posicionado no banco de dados Master.

Sintaxe

xp_cmdshell 'nome_arquivo'

Onde:

nome_arquivo é o nome do programa que deseja executar.

Exemplo:

```
xp_cmdshell 'calc.exe'
```

Para mais informações consulte SQL Server Books Online em Transact-SQL ítem Stored Procedures ou no Help existe tópicos para cada procedimento.

Componentes do Banco de Dados

Um banco de dados é composto de *objetos*, *índices*, *tipos de dados* e *restrições:* Cada objeto tem uma linha correspondente na tabela sysobjects. Seu tamanho mínimo é 1 Mb.

Objetos

Um objeto contém dados ou interage com os dados. Cada objeto tem uma linha correspondente a tabela sysobjects.Existem os seguintes tipos de objetos:

Tabela [table]	Conjunto de linhas, compostas de colunas. Cada coluna armazena um item de dado.
Visão [view]	Uma forma alternativa de visualizar dados em uma tabela ou mais.
Default	Um valor que é inserido numa coluna caso não tenha sido informado um valor.
Regra [rule]	Valida os dados que podem ser inseridos em uma coluna.
Procedimento armazenado	Uma seqüência de comandos SQL, compilados e armazenados

[stored procedure]	no banco de dados.
Gatilho [trigger]	Uma seqüência de comandos executados automaticamente quando os dados são modificados numa tabela.

Índices

Um índice é composto de ponteiros para os dados, ordenados pelo valor de uma ou mais colunas. Através de um índice, é possível acessar mais rapidamente os dados, dado o valor de algumas colunas.

Tipos de dados

Um tipo de dados especifica quais os valores que podem ser armazenados em uma coluna.

Restrições [constraints]

Uma restrição reforça a integridade dos dados em uma tabela, ou entre duas tabelas, controlando quais dados podem ser inseridos.

Estrutura dos bancos de dados

Todo banco de dados do SQL Server é constituído de dois ou mais arquivos físicos de sistema operacional. Podem haver três tipos diferentes de arquivos físicos:

- Arquvios primários: Todo banco de dados inclui ao menos um arquivo primário, que é feito para armazenar todos os objetos de banco de dados, tais como tabelas e índices. Este arquivo também é usado para apontar para o resto dos arquivos que constituem o banco de dados.
- Arquivo secundário: Um banco de dados só terá um arquivo secundário se o arqvuivo primário não for grande o suficiente para aramzenar todos os dados. Um banco de dados pode ter um, ou muitos arquivos secundários.
- Arquivo de log: Todo banco de dados tem um arquivo de log, que é usado para rgistrar todas informações antes que elas sejam escritas em um arquivo primário ou secundário. Esses dados são utilizados para ajudar na recuperação, no caso de um problema com o banco de dados. Um banco de dados pode ter um ou vários arquivos de log se o arquivo de log original ficar sem espaço.

Nomes lógicos e físicos

Um dispositivo de banco de dados é um arquivo do sistema operacional, por exemplo, o banco de dados MASTER é o arquivo C:\MSSQL7\DATA\master.mdf, localizado no servidor. Como default, os bancos de dados são criados no diretório *C:\MSSQL7*\DATA, mas podem ser criados em qualquer drive ou diretório acessível ao SQL Server.
Cada banco de dados tem um *nome físico* (o caminho e nome do arquivo) e um *nome lógico* (nome usado dentro do SQL Server). Os dois não precisam estar relacionados. Por exemplo, ao criar um banco de dados chamado VENDAS, o SQL Server, por default, cria um arquivo chamado VENDAS.mdf, no diretório *C:\MSSQL7*\DATA.
O nome físico pode ser qualquer nome suportado pelo sistema operacional. O nome lógico pode ter até 30 caracteres e geralmente só contém letras e números (espaços não são recomendados).

Subdivisões de espaço

Cada banco de dados é dividido ainda em:
Unidades Cada unidade de alocação tem 512 Kb (meio megabyte). Um banco de dados

de alocação:	ocupa sempre um número inteiro de unidades de alocação.
Extents:	Um *extent* tem 64 Kb. Cada objeto de banco de dados (tabelas, índices) ocupa um número inteiro de *extents*
Página [page]:	Uma página (8Kb) é a unidade mais básica de armazenamento. Um objeto do banco de dados sempre cresce em páginas e em alguns casos pode ficar fragmentado, disperso em páginas distantes uma da outra.

Arquivos predefinidos

Ao instalar o SQL Server, são criados quatro bancos de dados, com os seguintes nomes lógicos:
MASTER: composto pelos arquivos *master.mdf* e *mastlog.ldf* (arquivo primário e de log)
MSDB: composto pelos arquivos *msdbdata.mdf* e *mastlog.ldf* (arquivo primário e de log, que por sinal é o mesmo arquivo de log do banco de dados *master*)
MODEL: composto pelos arquivos *model.mdf* e *modellog.ldf*.
TEMPDB: composto pelos arquivos *tempdb.mdf* e *templog.ldf*

5 - Criando Bancos de Dados

Objetivos:
- Aprender a gerenciar bancos de dados, criando, alterando ou excluindo-os com o Enterprise Manager ou com comandos SQL;
- Aprender a criar tabelas e alterar sua estrutura.

Gerenciando Bancos de Dados

Você cria um banco de dados [database] definindo o seu nome, nome do arquivo, tamanho inicial, tamanho máximo e taxa de crescimeno. Inicialmente apenas o administrador do sistema (SA) pode criar, modificar o tamanho e excluir os bancos de dados, mas ele pode conceder

permissões a outros usuários para isso. Em versões anteriores do SQL Server, havia a a necessidade de criação de dispositivos [devices], e dentro destes deviam ser criados os bancos de dados. Bem, não existem mais dispositivos no SQL Server 7.0.
Depois de fazer excluir ou fazer alterações no do banco de dados é recomendável fazer backup do banco de dados **master**, porque ele contém informações sobre cada banco de dados. Mais especificamente, a tabela de sistema *sysdatabases*, do banco de dados master, armazena as informações sobre todos os bancos de dados. Mais adiante, discutiremos com mais detalhes as tabelas de sistema.

O Log de Transações [Transaction Log]

Quando você cria um banco de dados, é criado também um *log de transações* [transaction log] para esse banco de dados. Esta é uma área reservada onde todas as alterações feitas no banco de dados são registradas. Qualquer comando SQL que modifica os dados registra as alterações *antes* no log de transações, *depois* nas tabelas alteradas.
Quando é executado um comando que altera os dados (insere, altera ou exclui linhas numa tabela), essa alteração é salva primeiro no log de transações, escrevendo diretamente em disco. Os dados são alterados apenas em memória.
Periodicamente, o SQL Server faz um *checkpoint*, um processo que grava em disco as alterações feitas em memória. (Geralmente um checkpoint é feito uma vez por minuto).
O log de transações permite recuperar o banco de dados a um estado consistente, em caso de uma pane no sistema. Sempre que o SQL Server inicia, ele verifica o log de transações para saber se alguma alteração foi iniciada, mas não salva nos dados. As transações que não foram confirmadas (*commited*) são canceladas.
Pode ser interessante colocar o log fisicamente em um disco diferente dos dados, o que melhora o desempenho, pois as operações de E/S podem ser feitas simultaneamente nos dois.
Por padrão, ao se criar um banco de dados, o log de transações é criado com 25% do tamanho do banco de dados. Pode-se mudar o tamanho do mesmo. Recomenda-se alocar para o log de 10 a 25% do tamanho do banco de dados.

Criando bancos de dados com o Enterprise Manager

Vamos criar um banco de dados com o SQL Enterprise Manager.
No SQL Enterprise Manager, conecte-se ao servidor desejado. Clique em "Databases" com o botão direito e selecione **New Database...**
Em "Name" coloque Exemplo. Note que ao definir o nome, o nome do arquivo [File Name] (que não é necessariamente o mesmo nome definido em "Name", apesar de poder ser), muda para o nome que você está digitando seguido de _data. Se você clicar na guia Transaction Logs, verá que o arquivo de log está sendo criado, com o nome do banco de dados seguido de _log, e no caminho definido para o banco de dados.
Em "Location", o local no disco onde você quer armazená-lo. Por padrão, o SQL Server define o local como sendo a subpasta \DATA, na pasta de instalação do SQL Server. O tamanho usado, "Size", aparece como 1 Mb por default, substitua esse valor por 10 Mb. O tamanho alocado para o log, por default, é também 1 Mb, como você pode ver clicando na guia Transaction Logs. Mude o tamanho para 2Mb.
A opção "Automatically grow file" determina se o arquivo poderá ser expandido à medida que for ficando cheio. Se essa opção estiver marcada, você pode determinar a taxa de crescimento (a cada vez que ele for expandido, será expandido em N% ou N megabytes) do arquivo [File growth] em porcentagem ou megabytes. Também é possível determinar um tamanho limite para o arquivo [Restrict file growth] ou deixá-lo crescer indeterminadamente [Unrestricted file growth].

Por padrão, como se percebe, o banco de dados tem como propriedades crescer automaticamente, em incrementos de10 por cento, e sem limite de crescimento.
Clique em **Ok** e aguarde alguns instantes: o banco de dados será criado, com 10 Mb para os dados e 2 Mb para o log de transações.
Nota: O arquivo de banco de dados é gravado com a extensão **.mdf** ou **.ndf**, dependendo se ele for um arquivo primário ou não-primário no banco de dados. Veremos isso melhor em Grupos de Arquivos. O arquivo de log é gravado com a extensão **.ldf**.
Nota: Ao ser criado, um banco de dados é uma cópia do banco de dados **model**. Quaisquer opções ou configurações do banco de dados **model** são copiadas no novo banco de dados.

Criando bancos de dados com comandos SQL

Para gerenciar os bancos de dados com comandos SQL é necessário que se esteja posicionado no banco de dados **master**.
Você também pode criar um banco de dados com o comando SQL, CREATE DATABASE.
Sintaxe

```
CREATE DATABASE nome_bancodedados
[ON {
[PRIMARY] (NAME = nome_lógico_arquivo,
    FILENAME = 'caminho_e_nome_arquivo'
    [, SIZE = tamanho]
    [, MAXSIZE = tamanho_máximo]
    [, FILEGROWTH = taxa_crescimento]
    }[,...n]
]

[LOG ON
    {
    (NAME = nome_lógico_arquivo,
    FILENAME = 'caminho_e_nome_arquivo'
    [, SIZE = tamanho])
    }[,..n]
]
```

Onde:
nome_bancodedados é o nome do banco de dados que se deseja criar.
nome_logico_arquivo é um nome usado para referenciar o arquivo em quaisquer comandos SQL executados depois que o banco de dados tiver sido criado.
PRIMARY: Esta opção especifica o grupo de arquivos primário. O grupo de arquivos primário deve contre todas as tabelas de sistema para o banco de dados. Um banco de dados só pode ter um grupo de arquivo PRIMARY. Se não for especificado algum, o primeiro listado será o primário. (Veremos grupos de arquivos em breve).
FILENAME: Aqui deve-se especificar o caminho e nome do arquivo que você está criando. O arquivo deve estar localizado na mesma máquina que o servidor SQL Server. Ele pode estar em uma unidade de disco diferente contanto que esteja na mesma máquina.
SIZE: Especifica o tamanho em megabytes que você quer alocar para o seu banco de dados. O valor mínimo é 1MB, e o padrão é 3MB para arquivos de dados, e 1MB para arquivos de log. (Obs.: o padrão aqui diz respeito à criação do banco de dados por comandos SQL. Como vimos, quando criado pelo Enterprise Manager, o padrão é 1MB de tamanho tanto para arquivos de banco de dados quanto para arquivos de log).
MAXSIZE: Esta opção lhe permite especificar o tamanho máximo até o qual seu arquivo pode crescer. O padrão permite que seu arquivo cresça até que o disco esteja cheio.
FILEGROWTH: Especifica a taxa de crescimento do arquivo. Este ajuste não pode exceder a configuração de MAXSIZE. Um valor de 0 indica que não é permitido crescimento. O padrão é

10 por cento, significando que a cada vez que o arquivo cresce, será alocado um espaço adicional de 10 por cento para ele. Um banco de dados que esteja em mais de um arquivo (veja Arquivos para maiores detalhes), só é expandido depois que o último arquivo estiver cheio.

Na opção LOG ON se aplicam as mesmas definições acima, exceto pelo fato de não ser o arquivo de dados, mas sim o arquivo de log de transações que estará sendo criado.

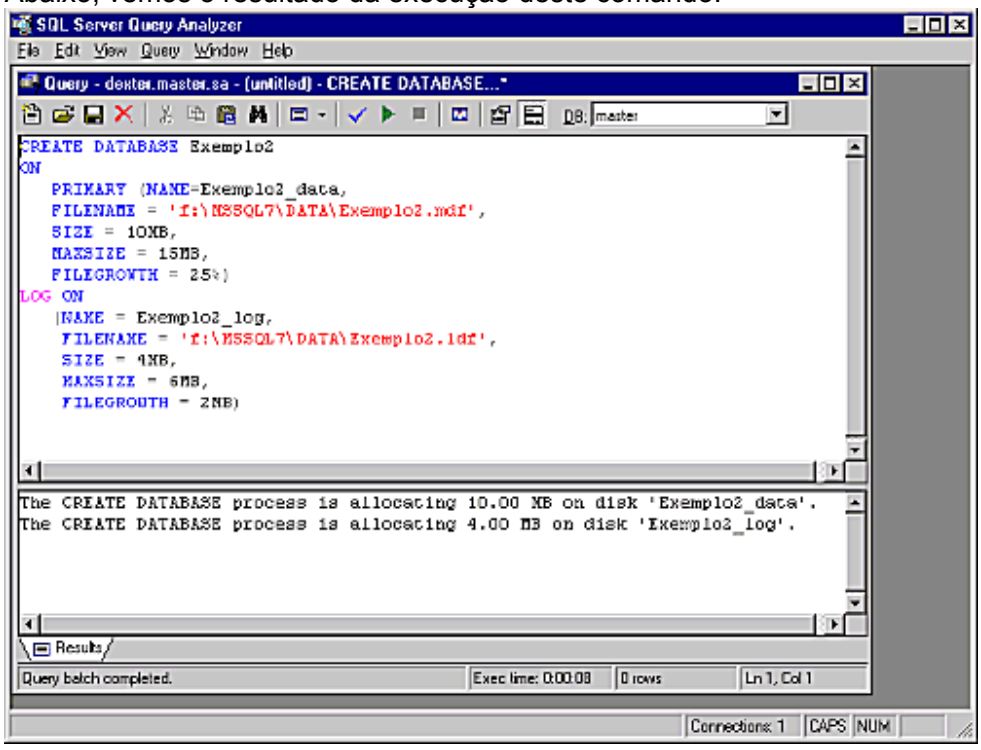
Caso LOG ON seja omitido, é criado um único arquivo de log com um nome gerado pelo sistema e um tamanho que seja 25 por cento da soma dos tamanhos de todos os arquivos de dados para o banco de dados.

Nota: A opção FOR LOAD pode ser adicionada antes da declaração LOG ON. Não recomenda-se usá-la, já que existe apenas para compatibilidade com versões anteriores. Esta opção coloca a opção **for dbo use only** como verdadeira, e o status como para carregar [*for load*]. Essa opção não é necessária, pois no SQL Server 7, temos o comando RESTORE pode recriar um banco de dados como parte de uma operação de recuperação.

Como exemplo, vamos criar um banco de dados, entrando com o seguinte código SQL no Query Analyzer.

```
CREATE DATABASE Exemplo2
ON
   PRIMARY (NAME=Exemplo2_data,
   FILENAME = 'C:\MSSQL7\DATA\Exemplo2.mdf',
   SIZE = 10MB,
   MAXSIZE = 15MB,
   FILEGROWTH = 25%)
LOG ON
   (NAME = Exemplo2_log,
    FILENAME = 'C:\MSSQL7\DATA\Exemplo2.ldf',
    SIZE = 4MB,
    MAXSIZE = 6MB,
    FILEGROWTH = 2MB)
```

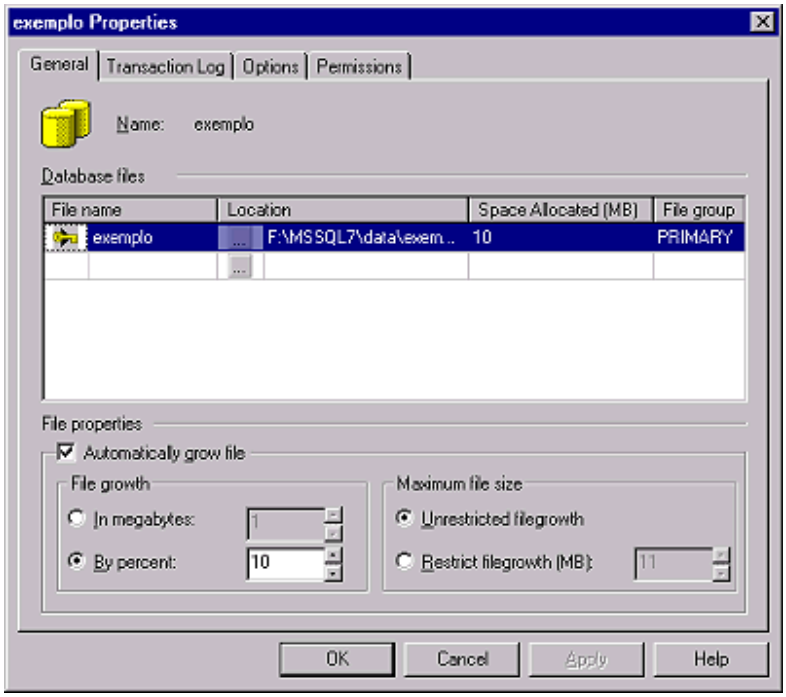
Abaixo, vemos o resultado da execução deste comando:



Criando um banco de dados com o assistente de criação de banco de dados

O assistente de criação de banco de dados te encaminha através dos passos necessários para a criação de um banco de dados e de um log. Para iniciá-lo, faça o seguinte:

1. A partir do Enterprise Manager, com algum servidor selecionado, selecione Tools | Wizards. Todos os objetos que têm assistentes são listaos em ordem alfabética.
2. Depois que você selecionar Databases (bancos de dados) a árvore deve se expandir para lhe mostrar uma lista dos assistentes disponíveis para bancos de dados.
3. Selecione Create Database Wizard. Aparece a tela de boas-vindas. Clique em Next para continuar.
4. Escolha o nome do banco de dados, e o local onde ficarão seus arquivos de dados e de log. Clique em Next.
5. Mude o nome do arquivo e seu tamanho inicial, conforme desejar.
6. Por fim aparecem as opções, que são não permitir o crescimento do banco de dados [Do not automatically grow the database]. Caso você escolha por crescer automaticamente o banco de dados [Autmotaically grow database files], você tem as opções de incrementá-lo em megabytes [Grow the files in Megabytes], ou porcentagem [Grow the files by percent], e definir as taxas de incremento. Por fim, você pode limitar o tamanho máximo do arquivo [Restrict file growth to (MB)] ou deixá-lo crescer o quanto for necessário [Unrestricted File Growth]. Faça suas opções e clique em Next para continuar.
7. Então, defina o nome do arquivo e tamanho inicial para o arquivo de log. Clique em Next para continuar.
8. Finalmente, defina o mesmo citado no item 6 acima, para o arquivo de log. Clique em Next.
9. Aparece a tela final, mostrando-lhe o que você definiu. Se estiver satisfeito, clique em Finish para criar o banco de dados. Caso contrário, clique em Back e altere as opções que achar necessário.

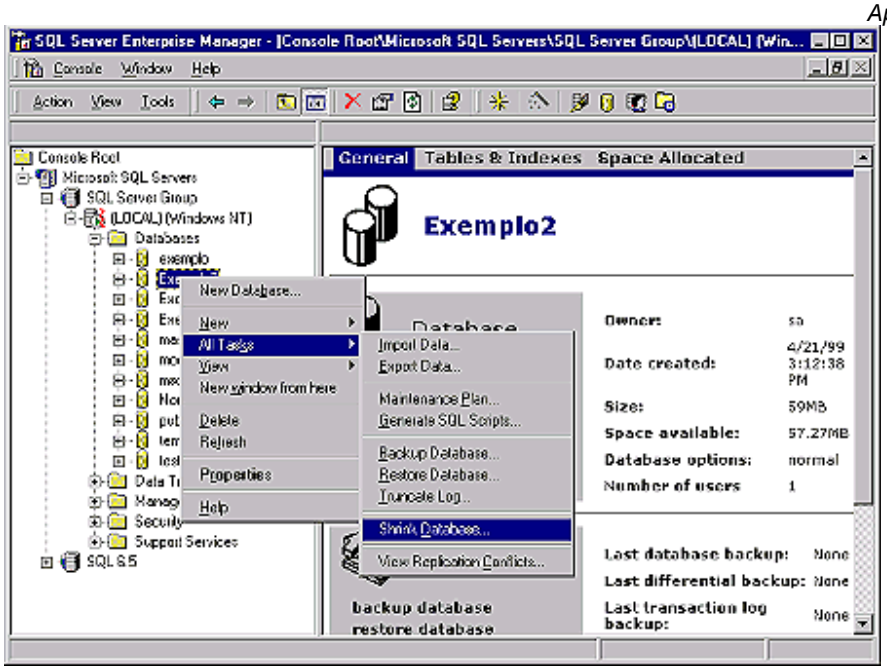


Alterando um banco de dados pelo Enterprise Manager

Após criar um banco de dados, você pode aumentá-lo ou reduzí-lo. No Enterprise Manager, clique no banco de dados com o botão direito e em **Properties**. Você verá os tamanhos atuais dos dados e do log e quanto espaço disponível existe em cada um.

Para aumentar o tamanho do banco de dados, informe o novo tamanho em "Space Allocated" (que deve ser maior que o tamanho original). Note que você só pode alterar, na guia General, o espaço alocado para o banco de dados, e as opções de crescimento [File Properties]. É só clicar em Ok ou em Apply que as mudanças são aplicadas.
Vamos aumentar o tamanho do banco de dados de 10 para 11 Mb. Basta para isso, digitar 11 em "Space Allocated", e clicar em Ok.
Para reduzir um banco de dados pelo Enterprise Manager, selecione o banco de dados que se quer reduzir, clique com o botão direito nele, selecione All Tasks | Shrink database, como mostrado abaixo

Na janela que aparece a seguir, você tem algumas opções, que são:



- Reorganize database: esta opção, se selecionada, faz uma espécie de desfragmentação do banco de dados, agrupando as páginas preenchidas, e deixando as páginas livres no fim do arquivo.
- Shrink database files: esta opção disponnibiliza o espaço livre que houver no fim do arquivo, para o sistema operacional.

Alterando um banco de dados com comandos SQL

Para expandir o banco de dados, pode-se usar ALTER DATABASE. Para utilizar este comando, deve-se estar posicionado no banco de dados **master**:

Sintaxe

```
ALTER DATABASE nome_bancodedados
 MODIFY FILE
  (NAME = nome_logico_arquivo,

   SIZE = novo_tamanho
  )
```

Onde:
nome_bancodedados é o nome do banco de dados que se deseja alterar.
nome_logico_arquivo é o nome lógico dado ao arquivo na sua criação.
novo_tamanho é o tamanho novo do banco de dados. Esse tamanho é fornecido em MegaBytes.
Por exemplo:
Vamos fazer o banco de dados aumentar para 12 Mb. Para isso, digite:

```
ALTER DATABASE Exemplo2
MODIFY FILE
  (NAME = Exemplo2_data,
   SIZE = 12MB
  )
```

Para expandir o log de transações do banco de dados Exemplo2 para 3MB, faça:

```
ALTER DATABASE Exemplo2
```

```
MODIFY FILE
  (NAME = Exemplo2_log,
   SIZE = 3MB
  )
```

Com o comando SQL ALTER DATABASE, também é possível acrescentar arquivos ao banco de dados. Para isso, usa-se a opção ADD FILE. Vamos adicionar um arquivo secundário ao banco de dados Exemplo2, com 4MB iniciais, e tamanho máximo de 7MB. Veja mais sobre arquivos secundários.

```
ALTER DATABASE Exemplo2
   ADD FILE (NAME='Exemplo2_dados2',
   FILENAME='C:\mssql7\data\Exemplo2.ndf',
   SIZE=4MB,
   MAXSIZE=7MB)
```

Para reduzir o tamanho de um banco de dados, usa-se o comando DBCC SHRINKDATABASE. Este comando reduz o tamanho de todos os arquivos de dados no banco de dados mas não diminui o tamanho dos arquivos de log.

Sintaxe

```
DBCC SHRINKDATABASE
  (Nome_BancodeDados ,porcentagem_final
  [, {NOTRUNCATE | TRUNCATEONLY}]
  )
```

Onde:

nomebancodedados é o nome do banco de dados que se deseja alterar o tamanho.
porcentagem_final é a porcentagem de espaço livre a ser deixada no banco de dados depois que o mesmo for reduzido.
NOTRUNCATE faz com que o espaço liberado permaneça nos arquivos de banco de dados. Se não especificado, o espaço liberado é tornado disponível para o sistema operacional.
TRUNCATEONLY faz com que o espaço não utilizado em arquivos de dados seja liberado para o sistema operacional e encolhe o arquivo até o último espaço utilizado, reduzindo o tamanho do arquivo sem mover quaisquer dados. Nenhuma tentativa é feita para relocar colunas em páginas não-alocadas. *porcentagem_final* é ignorado quando TRUNCATEONLY for utilizado.
Não se pode diminuir o tamanho do banco de dados para um tamanho menor do que o tamanho mínimo do arquivo, que é espcificado quando o arquivo foi originalmente criado.
Nota: O banco de dados não pode ficar menor que o tamanho do banco de dados **model**.
Pode-se também reduzir todo o banco de dados usando o comando DBCC SHRINKFILE. Este comando reduz o tamanho de um arquivo de dados específico em um banco de dados.

Sintaxe

```
DBCC SHRINKFILE ({nome_arquivo|id_arquivo}, [novo_tamanho][,
  {EMPTYFILE | NOTRUNCATE | TRUNCATEONLY}])
```

Todas as opções já foram explicadas para o comando DBCC SHRINKDATABASE, com duas exceções:

- Aqui se especifica o nome do arquivo, e não do banco de dados.
- O novo tamanho é dado em MB, não em porcentagem como acima.
- A opção EMPTYFILE migra todos os dados do arquivo especificado para outros arquivos no mesmo grupo de arquivos. O SQL Server não permite mais que dados sejam colocados no arquivo em que foi utilizada a opção EMPTYFILE. Esta opção permite que o arquivo seja excluído com o comando ALTER DATABASE, com a opção REMOVE FILE.

Nota: Quando se usa qualquer das opções EMPTYFILE, NOTRUNCATE ou TRUNCATEONLY, não se especifica o novo tamanho.

Como exemplo, vamos reduzir o arquivo secundário do banco de dados Exemplo2 (Exemplo2.ndf), que chamamos de Exemplo2_Dados2 para 2MB. Para isso, entre com o seguinte comando no Query Analyzer:

```
DBCC SHRINKFILE (Exemplo2_Dados2,2)
```

O banco de dados a ser reduzido não precisa estar em modo de único usuário, outros usuários podem estar trabalhando com o banco de dados quando ele é reduzido.

Excluindo um banco de dados pelo Enterprise Manager

Para excluir um banco de dados no Enterprise Manager, clique no Banco de Dados e com o botão direito , clique em **Delete**. Confirme a exclusão. Após excluir um banco de dados, não é possível recuperar os dados, a não ser que você tenha feito um backup. Como iremos usar este banco de dados nos exemplos posteriores, crie-o novamente com as mesmas características que foi criado anteriormente.

Excluindo um banco de dados com comandos SQL

Para excluir um banco de dados, usa-se:

Sintaxe

```
DROP DATABASE nome_bancodedados[, nome_bancodedados...]
```

Onde:

nome_bancodedados é o nome do banco de dados que se deseja excluir.

Você pode excluir um banco de dados ou múltiplos banco de dados.

Exemplo:

Para excluir vários banco de dados ao mesmo tempo , usa-se:

```
DROP DATABASE Exemplo, Exemplo2
```

Se quiséssemos ter excluido somente o banco de dados Exemplo2, teríamos usado:

```
DROP DATABASE exemplo3
```

Se executar este comando crie o banco de dados Exemplo novamente, pois iremos usá-los nos exemplos posteriores.

Arquvos e grupos de arquivos

Arquivos e grupos de arquivos são a nova estrutura de armazenamento do SQL Server. Um banco de dados é armazenado em uma estrutura de arquivos, e então um grupo de arquivos padrão é criado quando você cria seu banco de dados. Outros arquivos podem ser acrescentados ao seu projeto. Estes arquivos podem ser agrupados com base em grupos de arquivos definidos pelo usuário. Na maioria dos casos, você usará apenas o grupo de arquivos padrão, e ter apenas um arquivo para a sua estrutura de banco de dados. Como veremos,

entretanto, assim que você estiver familizarizado com o conceito de arquivo e grupo de arquivo, o desempenho pode ser aumentado e a administração tornada mais fácil.
O uso de grupos de arquivos é uma técnica avançada de projeto de banco de dados. Você deve compreender a estrutura, transações, consultas, e dados de seu banco de dados profundamente de modo a determinar a melhor maneira de aramazenar tabelas w índices em grupos de arquivos específicos. Em muitos casos, o uso das capacidades de sistemas RAID (Redundant Array of Inexpensive Disks) fornece quase o mesmo ganho em desempenho que você poedrá obter com o uso de grupos de arquivos sem o encargo administrativo extra de definir e gerenciar grupos de arquivos.

Arquivos

Como dito anteriormente, o SQL Server cria bancos de dados e logs baseado em uma estrutura de arquivos ao invés da especificação de Dispositivo [Device] de versões anteriores. Isso permite que os arquivos de bancos de dados e de log sejam escalados com maior facilidade. Cada arquivo pode ser usado por apenas um banco de dados. Ele não pode ser compartilhado entre vários bancos de dados. Quando um banco de dados for excluído, seja através de DROP DATABASE ou do Enterprise Manager, o arquivo associado também é excluído.
O SQL Server tem três tipos de arquivos:

- **Primário.** O arquivo primário é o ponto de partida do banco de dados e aponta para o resto dos arquivos no banco de dados. Todo banco de dados deve ter ao menos um arquivo de dados primário. A extensão padrão e recomendada é **.mdf**.
- **Secundário.** Arquivos de dados secundários são os outros (não-primários) arquivos no banco de dados. Alguns bancos de dados podem não ter quaisquer arquivos secundários de dados, enquanto outros podem ter múltiplos arquivos secundários. A extensão padrão e recomendada é **.ndf**.
- **Log.** Um arquivo de log é uma área de armazenamento para todas as mudanças nos bancos de dados. Tdo banco de dados deve ter pelo menos um arquivo de log. A extensão recomendada é **.ldf**.

Nota: o SQL Server não te força a usar a extensão recomendada para os arquivos. Elas existem simplesmente para te ajudar a gerenciar seus arquivos de bancos de dados.

Grupos de arquivos

Os arquivos de bancos de dados são agrupados para fins de alocação e administração. Alguns sistemas podem ter seu desempenho aumentado pelo controle do armazenamento de dados e índices em unidades e disco específicas. Grupos de arquivos podem auxiliar nesse processo. O administrador de sistema pode criar grupos de arquivos para cada unidade de disco, e então definir que certas tabelas, índices, entre outros objetos, sejam armazenados em grupos de arquivos específicos.
Quando um banco de dados é criado, o grupo de arquivos primário contém o arquivo primário. Outros grupos de arquivos (grupos de arquivos definidos pelo usuário) podem ser criados e agrupados para fins de alocação e administração. Você cria um grupo de arquivos como uma coleção nomeada de arquivos. Nenhum arquivo pode ser membro de mais de um grupo de arquivos.
Existem três tipos de grupos de arquivos

- **Primário.** O grupo de arquivo primário contém o arquivo de dados primário e quaisquer outros arquivos de dados não atribuídos a outros grupos de arquivos. Todas as tabelas de sistema são armazenadas no grupo de arquivos primário.

- **Definido pelo usuário [User-defined].** Grupos de arquivos definidos pelo usuário são criados usando a palavra-chave FILEGROUP quando se utiliza os comandos CREATE DATABASE ou ALTER DATABASE.
- **Padrão.** O grupo de arquivos padrão contém todas as tabelas e índices que não têm um grupo de arquivo especificado quando eles são criados. Em cada banco de dados, apenas um grupo de arquivos de cada vez pode ser o grupo de arquivos padrão.

Nota: Arquivos de log nunca fazem parte de um grupo de arquivos. O espaço do log é gerenciado separadamente do espaço dos dados.

Vamos agora criar um grupo de arquivos chamado Grupo1_Exemplo2 no banco de dados Exemplo2, e então adicionar um arquivo secundário chamado Teste ao banco de dados Exemplo2, com o nome físico de TesteGrupo1_Exemplo2.ndf, com 3MB de tamanho, mas este arquivo será adicionado ao grupo Grupo1_Exemplo2. Entre com o seguinte código SQL:

```
ALTER DATABASE Exemplo2
ADD FILEGROUP Grupo1_Exemplo2
GO

ALTER DATABASE Exemplo2
ADD FILE
   (NAME=Teste,
    FILENAME='C:\MSSQL7\DATA\TesteGrupo1_Exemplo2.ndf',
    SIZE=3MB)
   TO FILEGROUP Grupo1_Exemplo2
GO
```

Note que o parâmetro TO FILEGROUP simplesmente especifica o grupo em que o arquivo criado deve ser adicionado.

Agora, vamos usar o comando CREATE DATABASE para criar um arquivo de dados primário, um grupo de arquivos definido pelo usuário, e um arquivo de log. Então, emitiremos o comando ALTER DATABASE para mudar o grupo de arquivos padrão para o grupo de arquivos definido pelo usuário.

```
USE MASTER
GO
-- dois traços indicam um comentário
-- CRIAR O BANCO DE DADOS

CREATE DATABASE ExemploNovo
ON
PRIMARY (NAME=ExemploNovo_data,
FILENAME = 'c:\mssql7\data\ExemploNovo.mdf',
SIZE=10MB,
MAXSIZE=15MB,
FILEGROWTH=10%),
   FILEGROUP ExemploNovo_FG1
      (NAME=ExemploNovo_FG1_DAT1,
       FILENAME='c:\mssql7\data\ExemploNovo_FG1_DAT1.ndf',
       SIZE=3MB,
       MAXSIZE=10MB,
       FILEGROWTH=10%),
      (NAME=ExemploNovo_FG1_DAT2,
       FILENAME='c:\mssql7\data\ExemploNovo_FG1_DAT2.ndf',
       SIZE=3MB,
       MAXSIZE=10MB,
       FILEGROWTH=10%)
   LOG ON
      (NAME=ExemploNovo_log,
       FILENAME='c:\mssql7\data\ExemploNovo.ldf',
       SIZE=5MB,
```

```
    MAXSIZE=15MB,
    FILEGROWTH=10%)
GO
-- Use ALTER para mudar o grupo de arquivos padrão
ALTER DATABASE ExemploNovo
MODIFY FILEGROUP ExemploNovo_FG1 DEFAULT
GO
```

Visualizando informações de arquivos e grupos de arquivos

Os seguintes procedimentos armazenados do sistema exibem informações sobre grupos de arquivos

- **sp_helpfile** [*file_name*]: exibe os nomes físicos e atributos dos arquivos associados com o banco de dados atual. Use este procedimento armazenado para determinar os nomes dos arquivos a serm anexados ou removidos do servidor.
- **sp_helpfilegroup** [*filegroup_name*]: exibe os nomes e atributos de grupos de arquivos associados com o banco de dados atual.

Criando Tabelas

Uma tabela [table] é um objeto do banco de dados, composto de zero ou mais *linhas* [rows], contendo os dados, organizados em uma ou mais *colunas* [columns]. Para criar a tabela, você pode usar o Enterprise Manager ou comandos SQL DDL (Data Definition Language - linguagem de definição de dados). Antes de criar suas tabelas, é importante levar em conta um bom projeto do banco de dados, que determina quais as informações a serem guardadas. Após criar as tabelas, você utiliza comandos SQL DML (Data Manipulation Language - linguagem de manipulação de dados) para inserir novas linhas numa tabela, alterar colunas das linhas existentes, excluir linhas e consultar dados.

Tipos de Dados

Cada coluna tem um *tipo de dados* [datatype], que determina que tipo de informação (caracteres, números, datas/horas) pode ser colocada na coluna e quais as características desses dados. O tipo é determinado quando a tabela é criada e não pode ser alterado posteriormente. Você pode usar *tipos de dados do sistema* [system datatypes], predefinidos, ou criar novos tipos de dados, chamados *tipos de dados do usuário* [user datatypes], baseados nos tipos preexistentes.
Os tipos de dados existentes são:

Para dados	**Tipo**	**Tamanho**
Caractere	*char*(n), *varchar*(n), *nvarchar*(n), *nchar*(n)	até *n* bytes
Numérico exato	*decimal*(p,e) ou *numeric*(p,e)	-depende-
Numérico aproximado	*float*, *real*	8, 4 bytes
Numérico inteiro	*int, smallint, tinyint*	4, 2, 1 byte
Monetário	*money, smallmoney*	8, 4 bytes
Data e hora	*datetime, smalldatetime*	8, 4 bytes
Binário	*binary*(n), *varbinary*(n)	*n* bytes
Texto e imagens	*text, image, ntext*	-variável-
Outros	*bit, timestamp*	1 bit, 8 bytes

Para dados contendo caracteres, *char*(n) armazena um número fixo de caracteres. Por exemplo, uma coluna do tipo *char(30)* tem sempre 30 caracteres. Se forem informados menos, o restante é completado com espaços. Já o tipo *varchar*(n) armazena uma quantidade variável de caracteres, até o máximo informado. Os tipos *nchar*(n) e *nvarchar*(n), armazenam dados Unicode, de comprimento fixo ou variável, e usam o conjunto de caracteres UNICODE UCS-2.

Os tipos "numéricos exatos", *decimal* e *numeric*, permitem armazenar dados exatos, sem perdas devidas a arredondamento. Ao usar esses tipos, você pode especificar uma *precisão*, que indica quantos dígitos podem ser usados no total e uma *escala*, que indica quantos dígitos podem ser usados à direita do ponto. Por exemplo, *decimal*(9,2) permite guardar 7 dígitos antes do ponto decimal e 2 após, num total de 9, assim o maior valor possível é 9999999,99.
Os tipos "numéricos inexatos", *float* e *real*, armazenam dados numéricos, mas nem sempre mantém a precisão suficiente para armazenar corretamente números de vários dígitos.
O tipo *money* é usado para valores monetários, ocupando 8 bytes em disco e permitindo valores entre -922.337.203.685.477,5808 e +922.337.203.685.477,5807 (922 trilhões). O tipo *smallmoney* permite valores entre - 214.748,3648 e +214.748,3647 (214 mil) e ocupa 4 bytes em disco.
Dos tipos inteiros, *int* usa 32 bits (4 bytes), permitindo armazenar até +/-2.147.483.647, *smallint* usa 16 bits (2 bytes) permitindo +/-32767 e *tinyint* usa 8 bits (1 byte), permitindo números não-negativos de 0 a 255.
O tipo *datetime* armazena valores contendo a data e hora, com precisão de 1/300 de segundo, entre 1º de janeiro de 1753 e 31 de dezembro de 9999 (o século é sempre armazenado). O tipo *smalldatetime* ocupa menos espaço e armazena datas e horas de 1º de janeiro de 1900 até 6 de junho de 2079, com precisão de 1 minuto.
Tipos binários são usados para dados que o SQL Server não interpreta, por exemplo, o conteúdo de um arquivo binário. O tipo *text* é usado para colunas com dados "memo", ou seja, com texto de tamanho variável; o tipo *ntext* armazena dados Unicode de tamanho variável. O tipo *image* armazena imagens, também de tamanho variável.
Os tipos *text* e *ntext*, armazenam dados de tamanho variável, mas podem armazenar 1.073.741.823 caracteres, para o caso do *ntext*, e 2.146.483.647 caracteres para o caso do tipo *text*. Enquanto isso, os tipos *varchar* e *nvarchar* armazenam "somente" 8000 caracteres (*varchar*) ou 4000 caracteres (*nvarchar*)
O tipo *bit* armazena valor 1 ou 0. Uma coluna do tipo *timestamp* não pode ser alterada pelo usuário. Ela é definida automaticamente com a data e hora atual quando a linha é inserida ou atualizada.

Definindo novos tipos de dados

Você pode criar seus próprios tipos de dados, para facilitar a padronização, usando o procedimento de sistema *sp_addtype* ou o Enterprise Manager.
Usando o procedimento de sistema *sp_addtype:*
Sintaxe

```
sp_addtype nome_tipo, tipo_dado [, valor_null]
```

Onde:
nome_tipo é o nome do tipo de dado que deseja criar.
tipo_dado é o tipo de informação que ira conter este tipo criado. Exemplo: char, int,..etc.
valor_null identifica se este tipo pode ou não conter valores nulos.
Exemplos:
No Query Analyzer na lista "DB", selecione o banco de dados "Exemplo". Digite e execute os seguintes comandos:

```
sp_addtype cpf, 'char(11)'
go
sp_addtype nomepessoa, 'char(50)'
go
sp_addtype valorgrande, 'numeric(15,2)'
go
sp_addtype tipooperacao, 'SmallInt', NONULL
```

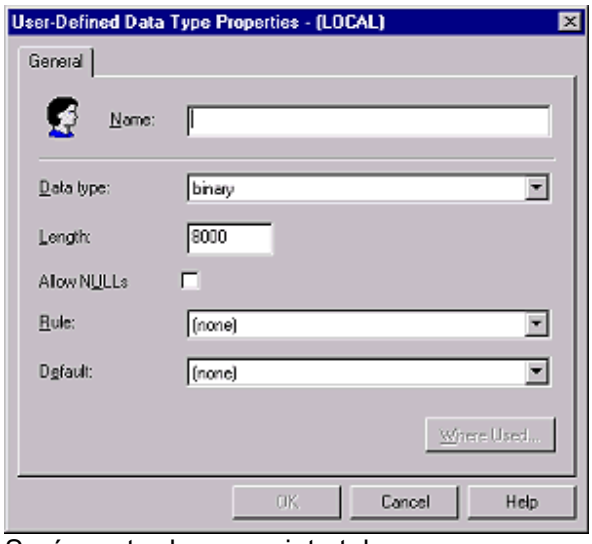
A palavra reservada **go** indica final de comando.
Com isso, usar o tipo 'cpf', por exemplo, é o mesmo que usar char(11), mas é mais intuitivo e fácil de entender. Se você especificar NONULL no tipo significa que ele não aceita valores nulos.
Você pode excluir um tipo com *sp_droptype*:
Sintaxe

```
sp_droptype nome_tipo
```

Onde:
nome_tipo é o nome do tipo de dados que deseja excluir.
Exemplo:
Para executar este exemplo continue posicionado no banco de dados Exemplo.

```
sp_droptype nomepessoa
```

Para criar, alterar ou excluir tipos com o Enterprise Manager, você deve abrir *Nome-do-banco-de-dados* clique em User Defined Datatypes com o botão direito clique em **Refresh** (para atualizar os dados) e com o botão direito clique em **New UserDefinedDataType...**.



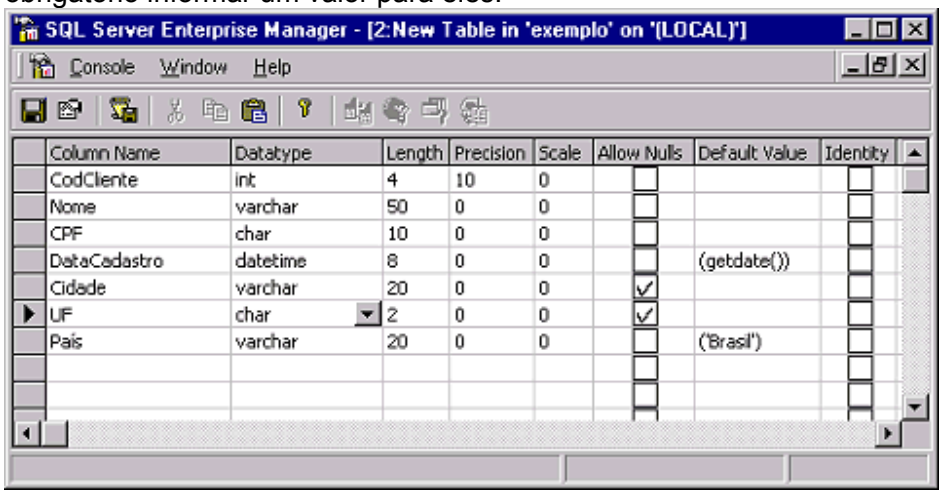
Será mostrada a seguinte tela:

'Name ' indica o nome do tipo de dados.
'Data type' é o tipo de informação que irá conter esse tipo criado.
'Length' indica o tamanho do tipo de dado.
'Allow Null' se esta opção estiver marcada indica que o tipo criado aceita valores nulos.
As opções 'Default' e 'Rule' permitem que você selecione uma regra ou um default, se houver algum, e ligue-o ao tipo de dados definido por você.
Para apagar algum tipo de dados, selecione-o do lado direito (quando você estiver com User Defined DataTypes selecionado do lado esquerdo do Enterprise Manager). Clique com o botão direito no tipo de dados que você quer excluir, e então selecione a opçao Delete.
Apague todos os tipos de dados e crie o tipo de dados Sexo. Em "Name" coloque dmSexo, "Data Type" selecione char, "Length" coloque 1. Mais tarde veremos como restringir o valor de um tipo.

Criando uma tabela com o Enterprise Manager

Um banco de dados pode ter no máximo 2 bilhões de tabelas e cada tabela pode ter no máximo 1024 colunas.
Para criar uma tabela com o Enterprise Manager, abra o banco de dados Exemplo. Dentro dele, selecione o item Tables. Clique em "Tables" com o botão direito e em **New Table**. Entre com o nome da tabela a ser criada. No caso Cliente; clique em Ok. Entre com os campos da tabela, conforme mostrado abaixo:
Veja que no título da janela, aparece o nome do banco de dados em que a tabela está sendo criada (no caso 'exemplo').

As colunas para as quais a opção "Nulls" está marcada permitem o valor NULL, ou seja, podem ser deixados sem preencher ao inserir dados. Já os outros são NOT NULL, ou seja, é obrigatório informar um valor para eles.



Column Name	Datatype	Length	Precision	Scale	Allow Nulls	Default Value	Identity
CodCliente	int	4	10	0			
Nome	varchar	50	0	0			
CPF	char	10	0	0			
DataCadastro	datetime	8	0	0		(getdate())	
Cidade	varchar	20	0	0	✓		
UF	char	2	0	0	✓		
Pais	varchar	20	0	0		('Brasil')	

Note que definimos algumas colunas com o tipo *char*, como CPF e UF, porque elas geralmente têm tamanho fixo. Já outras como Nome, Cidade e País, geralmente têm tamanho variável, por isso, para economizar espaço no banco de dados, usamos *varchar*.
A coluna Default especifica um valor default que é inserido caso nada tenha sido informado. No caso da data de cadastro, usamos a função getdate(), que retorna a data do dia. No caso do país, o default é a string "Brasil" caso nada seja informado.
Agora clique no botão "Save" para salvar a tabela.
Nota: Você pode ter um identificador exclusivo em todo o banco de dados, para uma coluna. Para isso, selecione o tipo de dados da coluna como **uniqueidentifier**, e marque a caixa de verificação IsRowGuid. Isso fará com que seja atribuído um valor default [Default Value] igual a (newid()). Veja mais sobre identificadores globalmente exclusivos.

Criando tabelas com comandos SQL

Uma tabela também pode ser criada com o comando CREATE TABLE do SQL. Por exemplo, a mesma tabela do exemplo anterior poderia ser criada com o comando abaixo, iremos mudar somente o nome da tabela e esta tabela será criada no banco de dados Exemplo:

```
CREATE TABLE Cliente1
(
CodCliente int NOT NULL,
```

```
Nome varchar(50),
CPF varchar(11) NULL,
DataCadastro datetime NOT NULL DEFAULT (getdate()),
Cidade varchar(20) NULL,
UF char(2) NULL,
País varchar(20) DEFAULT ('Brasil')
)
```

Note que a lista de colunas é delimitada com parênteses. Para cada coluna, deve-se informar NULL ou NOT NULL, indicando se esta permite valores nulos ou não. Caso essa opção não seja informada, como no caso de 'Nome' e 'País' acima, o SQL Server assume que a coluna é NOT NULL (geralmente).

Para saber se o SQL usa NULL ou NOT NULL por default execute o procedimento **sp_ dboption**. Se aparecer na primeira linha "ANSI null default' indica que o valor default é NULL, se não aparecer esta string a opção default é NOT NULL.

Sintaxe

```
sp_dboption nome_bancodedados 'ansi null default', opcao
```

Onde:

nome_bancodedados é o nome do banco de dados em que se deseja verificar a opção default.

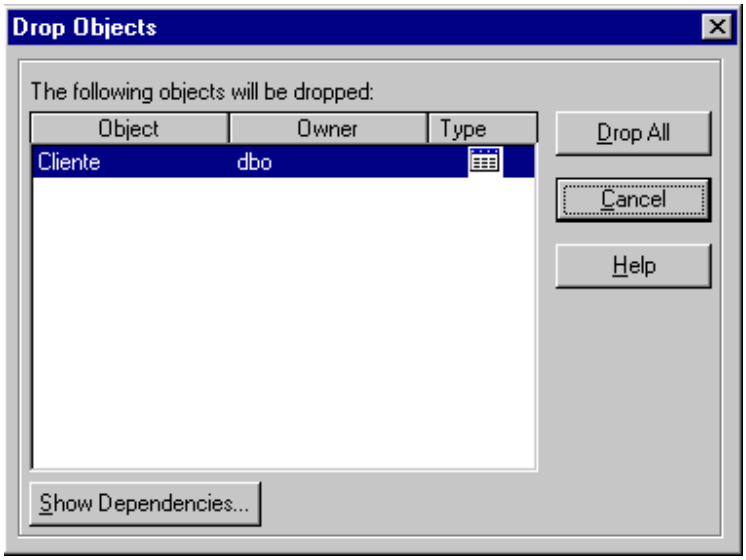
opcao se for true o valor default será NULL, se for false o valor default será NOT NULL.

Observação: O padrão ANSI utiliza Null como default, ao desativar esta opção no SQL, pode ser que esta opção na sua tabela não esteja de acordo com o que você esperava. Veja outros parâmetros de sp_dboption em "Definindo opções do banco de dados"

Execute os seguintes comandos para criar duas tabelas no banco de dados Exemplo:

```
CREATE TABLE Departamento
(
CodDepartamento int,
Nome varchar(50),
CodDeptSuperior int Null
)
go

CREATE TABLE Funcionario
(
CodFuncionario int,
Nome varchar(50),
CodDepartamento int,
Ramal int Null,
Salario money,
DataAdmissao datetime,
DataCadastro datetime NOT NULL DEFAULT (getdate()),
Sexo char(1)
)
```



Excluindo uma tabela

Para excluir uma tabela (chamado de *drop* no SQL Server) com o Enterprise Manager, clique na tabela com o botão direito, clique em **Delete** e pressione o botão Drop All .

Como iremos usar a tabela Cliente nos exemplos posteriores , crie-a novamente com a mesma estrutura definida anteriormente.
Através do SQL, pode-se usar o comando DROP TABLE.
Sintaxe:

```
DROP TABLE [[banco_dados.]Owner.]nome_tabela
[, [[banco_dados.]owner.]nome_tabela...]
```

Onde:
banco_dados é o nome do banco de dados a que a tabela pertence. Essa opção é opcional, ela será usada somente quando se estiver posicionado num determinado banco de dados e se deseja excluir a tabela de outro banco de dados.
nome_tabela é o nome da tabela que se deseja remover.
Este procedimento pode ser usado para excluir várias tabelas ao mesmo tempo.
Exemplos:

```
Drop Table Cliente1
```

ou

```
Drop Table Exemplo.dbo.cliente1
```

Alterando a estrutura das tabelas

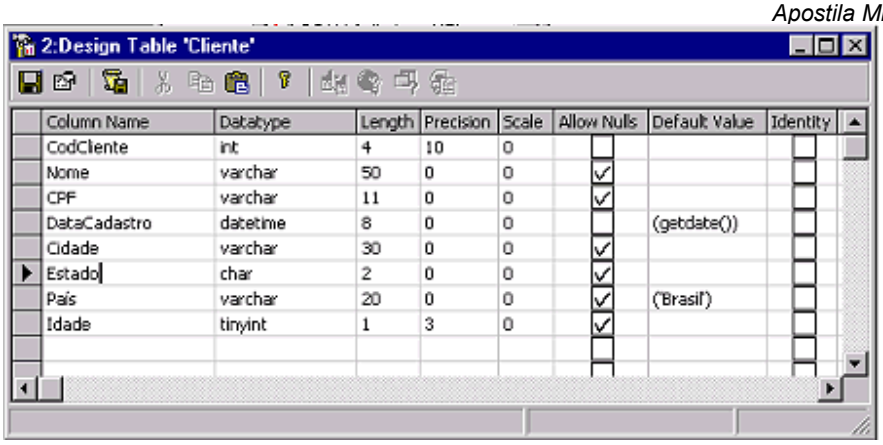
Depois que uma tabela for criada, pode-se mudar várias das opções que foram definidas qunado a tabela foi originalmente criada, incluindo:

- Colunas podem ser acrescentadas, modificadas ou excluídas. Por exemplo, o nome da coluna, comprimento, tipo de dados, precisão, escala, e o fato de aceitar ou não valores nulos, podem todos ser mudados, embora existam algumas restrições.
- Restrições PRIMARY KEY e FOREIGN KEY podem ser acrescentadas ou excluídas.
- Restrições UNIQUE e CHECK e definições DEFAULT podem ser acrescentadas ou excluídas.
- Uma coluna identificadora pode ser acrescentada ou removida usando a propriedade IDENTIY ou ROWGUIDCOL. A propriedade ROWGUIDCOL também pode ser adicionada ou removida de uma coluna existente, embora apenas uma coluna em uma tabela possa ter a propriedade ROWGUIDCOL de cada vez.

O nome ou o dono de uma tabela também podem ser modificados. Quando você faz isso, também deve-se mudar o nome da tabela em quaisquer gatilhos, procedimentos armazenados, scripts SQL, ou outro código de programação que utilize o nome ou proprietário antigo da tabela.
Nota: É importante considerar que a mudança de tipo de dados em uma coluna pode causar truncamento dos dados, ou mesmo ser impossível de ser feita (por exemplo, se você quiser converter um tipo char para um tipo inteiro e já houver valores não-numéricos armazenados nessa coluna).

Alterando a tabela com o Enterprise Manager

Essas modificações podem ser feitas no Enterprise Manager. Por exemplo, clique na tabela "Cliente" (dentro de "Exemplo\Tables", as tabelas do banco de dados Exemplo aparecerão do lado direito do Enterprise Manager) com o botão direito e clique em **Design Table**. No final da lista de colunas, acrescente uma nova coluna, com o nome "Idade", do tipo "tinyint" (idades não serão maiores que 255). Note que quando você adiciona uma nova coluna, a opção "Allow Nulls" deve ficar marcada.
Altere o comprimento de Cidade para 30. Também clique na coluna "UF" e altere o nome para "Estado". Após fazer isso, clique no botão "Save" para atualizar a tabela.



Alterando a tabela com comandos SQL

Também é possível alterar uma tabela com comandos SQL. Para isso, use o comando ALTER TABLE. Abaixo será mostrada uma sintaxe simples deste procedimento:

Sintaxe:

```
ALTER TABLE [banco_dados.[owner.]]nome_tabela

{
[ALTER COLUMN nome_coluna
   {novo_tipo_de_dados [(precisão[, escala])]
 | ADD {nome_coluna dados_coluna
 | [WITH CHECK | WITH NOCHECK]}
```

Onde:

banco_dados é o nome do banco de dados a que a tabela pertence. Essa opção é opcional, ela será usada somente quando se estiver posicionado num determinado banco de dados se e deseja excluir a tabela de outro banco de dados.

nome_tabela é o nome da tabela que deseja alterar.

nome_coluna é a coluna que se quer alterar.

novo_tipo_de_dados é o tipo de dados que a coluna aceitará a partir de agora.

WITH CHECK | WITH NOCHECK Especificam se os dados na tabela devem ou não ser validados contra uma nova ou reabilitada restrição FOREIGN KEY ou CHECK. Se não especificada, assume-se WITH CHECK para novas restrições e WITH NOCHECK para restrições reabilitadas. Veja mais sobre restrições.

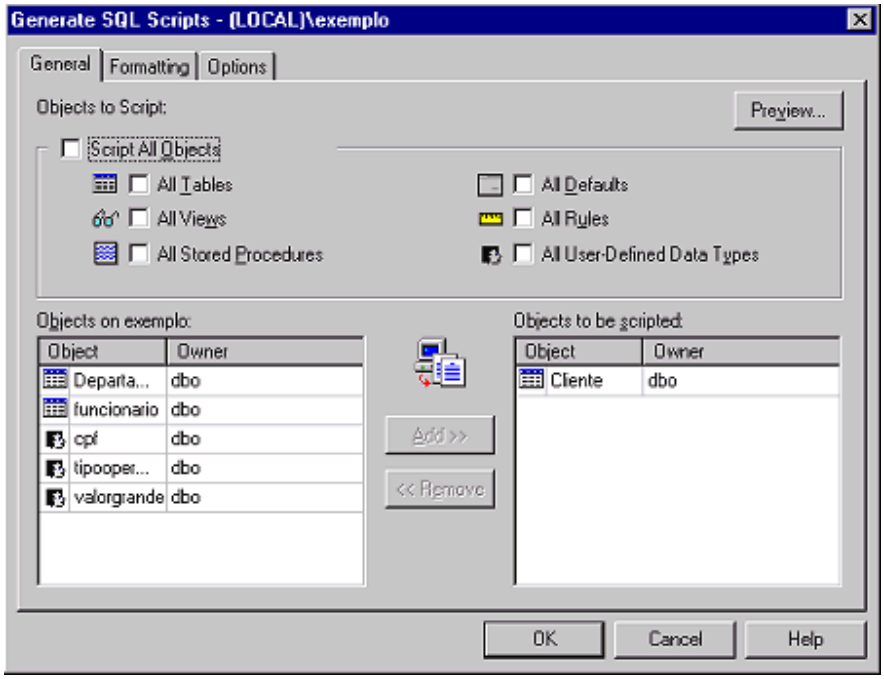
Exemplo:

```
ALTER TABLE Cliente
ADD ender varchar(50) NULL
GO
ALTER TABLE Cliente

ALTER COLUMN CIDADE VARCHAR (25)
```

Para renomear uma coluna, usa-se o procedimento *sp_rename*:

```
sp_rename 'Cliente.ender', Endereco
```



Nota: Perceba que ao executar o comando acima, você recebe um aviso que diz: "Cuidado: Mudar qualquer parte do nome de um objeto pode invalidar scripts e procedimentos armazenados.

Criando um "Script" das tabelas

Algumas tarefas são mais fáceis de fazer com os comandos CREATE TABLE. Por exemplo, para recriar um banco de dados em outro servidor, você pode salvar um arquivo (um *script*) contendo todos os comandos SQL CREATE TABLE usados para criar suas tabelas. Um script em geral é um arquivo contendo comandos SQL.
Se você não usou um comando SQL, pode fazer o próprio SQL Server gerar um script para você a partir da tabela existente. Para isso, no Enterprise Manager, clique no nome da tabela com o botão direito, e em "All tasks", e depois em "Generate SQL Scripts". Por enquanto, deixe as opções default. Clique em "Preview" para ver como ficam os comandos. Depois clique em "Save As..." para salvar o arquivo Script. Salve o arquivo com o nome Cliente (a terminaçao .SQL já é colocada por padrão).

Feche as janelas.

Para executar esse script em outro servidor ou outro banco de dados, basta abrir o SQL Query Analyzer (Iniciar | Programas | Microsoft SQL Server 7.0 | Query Analyzer), abrir o arquivo de script e executá-lo. Para abrir o arquivo Script clique em e procure o nome do arquivo. Note que você pode também modificar os comandos do script para criar tabelas com colunas ligeiramente diferentes.
Também é possível criar um script com todas as tabelas do banco de dados, ou com todos os objetos. Para isso, dê uma olhada em "Documentando a criação de bancos de dados" (veremos outros tipos de objetos mais tarde).

Definindo opções de bancos de dados

Uma porção de 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 serm 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 a 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 comparação 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 poe 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 nem 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.

***selec 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 frequentemente para desenvolvimento, fazendo com que o log de transações não fique cheio com tanta frequência. 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, frequentemente 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 tranquilidade e eficiência com bancos de dados, considere os seguintes fatos.

- Para obter melhor desempenho e segurança, aramazene 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 imdediatamente depois de cirar 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 cpaaz de modificar ou acessar seu banco de dados. Para evitar ficar sem espaço, faça o seguinte:
    - Aloque espaço suficiente para acomodar o crescimento.
    - Monitore frequentemente 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.

Exclusão de bancos de dados

Você não pode excluir bancos de dados que estejam:

- Atualmente aabertos para leitura ou escrita por outro usuário.
- Sendo restaurados.
- Publicando qualquer de suas tabelas (parte da replicação SQL).

Você também não pode excluir os seguintes bancos de dados:

- Master
- Model
- Tempdb

Embora lhe seja permitido excluir o banco de dados de sistema msdb, você não deve excluí-lo se usa ou pretende usar:

- Replicação
- SQL Server Agent
- Assistente de criação de páginas Web
- Histórico de backups
- Serviços de transformação de dados

Quando excluir um banco de dados, considere os seguintes fatos:

- Com o método SQL DROP DATABASE, você pode excluir vários bancos de dados de uma vez.
- O Enterprise Manager só lhe permite excluir um banco de dados de cada vez.
- Depois que você excluir um banco de dados, qualquer ID de login que usava o banco de dados excluído como seu banco de dados padrão, usará agora o banco de dados **master**.
- Você deve SEMPRE fazer backup do banco de dados **master**, sempre que qualquer novo banco de dados for adicionado ou excluído.

Documentação dos passos de criação de bancos de dados no SQL Server

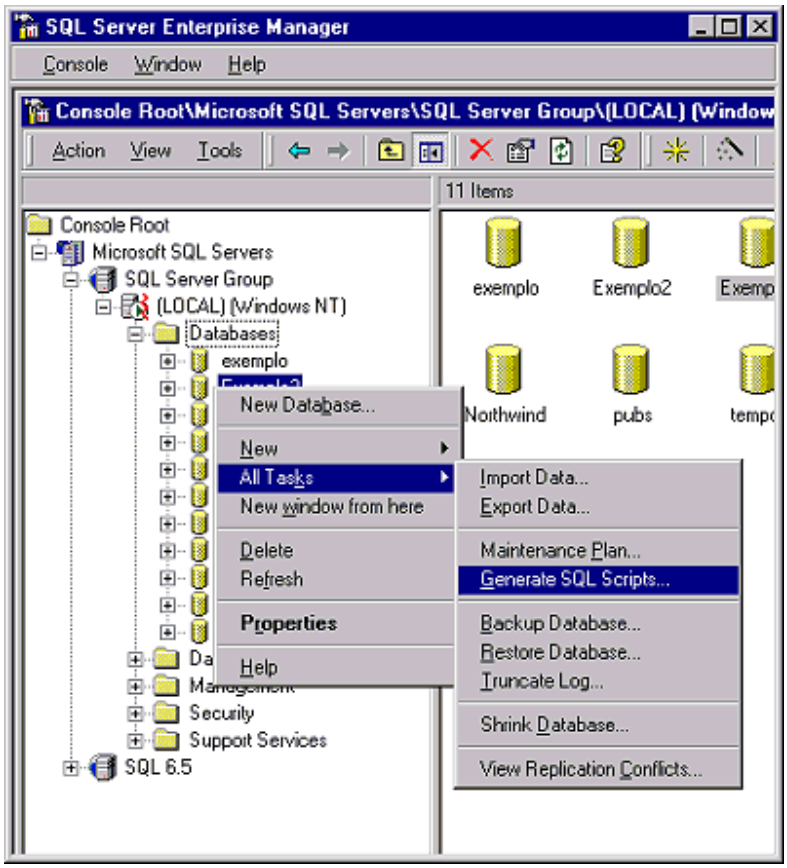
Documentar os passos de criação de bancos de dados SQL Server pode ser útil por diversas razões, mas é claro que a principal e mais motivadora é o fato de ter um back up do trabalho que você fez. Isso não necessariamente vai lhe prevenir de perdas de dados, mas vai salvar seu modelo, e um modelo de banco de dados é uma coisa terrível a se perder.
O SQL Server tem um gerador de script que torna fácil para você documentar, e se necessário reconstruir, seu banco de dados. O gerador de scripts pode construir o banco de dados e os objetos criados no banco de dados. Você tem a opção de selecionar desde todos até um único objeto. Você pode pegar um script de um banco de dados e rodá-lo em outro para criar cópias exatas de procedimentos armazenados, regras, gatilhos, etc. Você pode gerar scripts para os seguintes objetos:

- Tabelas [Tables]
- Procedimentos armazenados [Stored procedures]
- Gatilhos [Triggers]
- Índices [Indexes]
- Visões [Views]
- Usuários e Grupos [Users and Groups]
- Tipos de dados definidos pelo usuário [User-defined data types]
- Logins
- Regras [Rules]
- Default
- Tabelas-chave / DRI

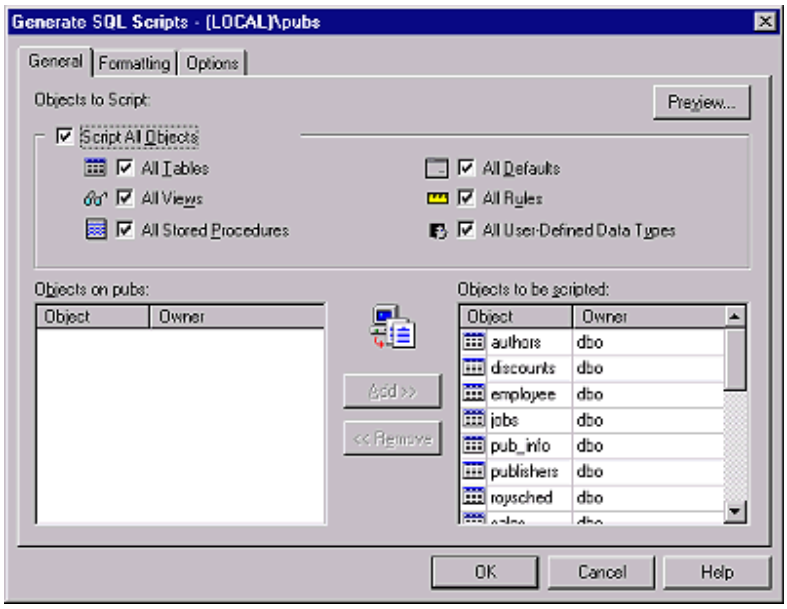
O esquema pode ser salvo em um arquivo único ou você pode querer dividí-lo baseado em objetos. Independentemente do seu método, você não tgem mais uma boa desculpa para um banco de dados não documentado.

Gerando um script a partir do Enterprise Manager

1. Expanda o grupo de servidores.
2. Expanda o servidor.
3. Expanda os bancos de dados.
4. Clique com o botão direito no banco de dados escolhido, e selecione All Tasks.
5. Selecione Gerar Scripts SQL [Generate SQL Scripts], como indicado abaixo.



6. Selecione os objetos que você deseja criar da janela que aparece a seguir, mostrada abaixo:



Você pode prever o arquivo primeiro ou simplesmente clicar em OK para salvá-lo em um arquivo.

6 - Consultando Dados

Visão Geral do Transact-SQL

A Sintaxe do SELECT

Manipulando Expressões

Condições de Pesquisa

Outros Recursos

Objetivos:
- Entender a divisão de comandos da linguagem Transact-SQL;
- Aprender a usar o comando SELECT e suas várias opções para fazer consultas aos dados.

A Sintaxe do SELECT

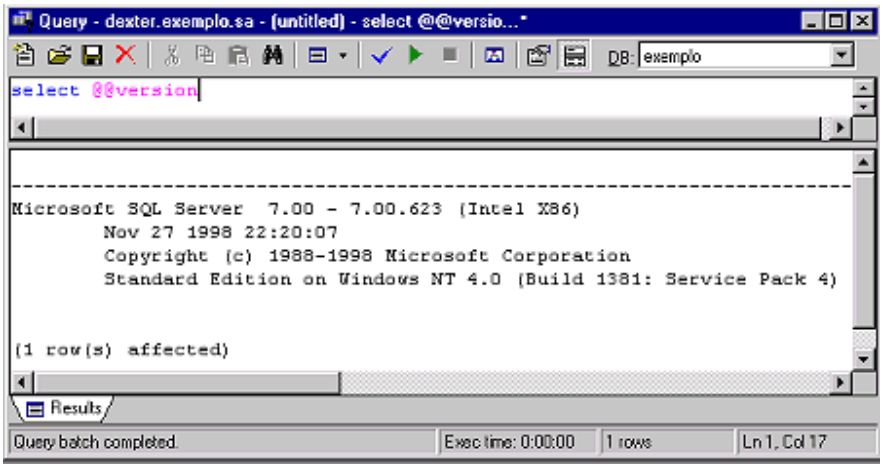
O comando SELECT recupera dados de uma ou mais tabelas. A sua sintaxe mais simples pode ser resumida da forma:

Sintaxe:
```
SELECT lista_de_colunas
FROM lista_de_tabelas
WHERE condições
```
Onde

A *lista_de_colunas* especifica quais colunas serão retornadas como resultado, separadas por vírgulas ou um asterisco (*) que indica todas as colunas da tabela.
A cláusula FROM, com uma *lista_de_tabelas*, especifica quais tabelas serão consultadas.
A cláusula WHERE especifica *condições* que devem ser satisfeitas pelas linhas das tabelas.
O comando Select pode ser utilizado para mostrar o conteúdo de variáveis, valores literais, etc...
Como exemplo execute o seguinte comando no Query Analyzer:

```
Select @@VERSION
```

O resultado será:



@@Version é uma variável global do SQL Server que contém a versão do SQL Server utilizado.
Para mostrar um valor literal digite o comando:

```
Select 'Teste'
```

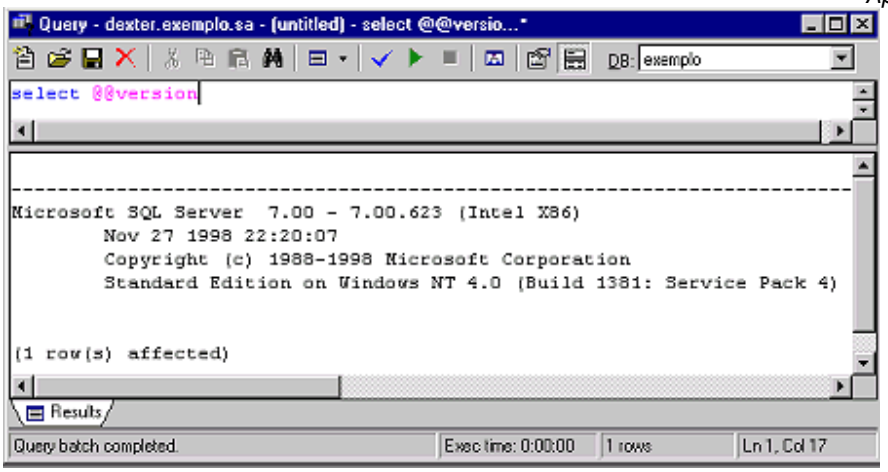
O resultado será a palavra Teste.

Exemplo: Consultando Todas as Colunas

Para nossos exemplos, vamos usar o banco de dados *pubs*, um banco de exemplo que é instalado pelo SQL Server. Esse banco de dados armazena informações sobre livros, na tabela *titles* e sobre autores de livros, na tabela *authors* (entre outras coisas).
Execute o Query Analyzer e se conecte ao servidor como "sa". Na lista "DB", selecione "pubs".
Digite e execute o seguinte comando:

```
select * from authors
```

O resultado irá mostrar todas as colunas e todas as linhas da tabela 'authors' (ou seja, todo o seu conteúdo), como abaixo (algumas linhas e colunas foram omitidas):



Note a mensagem "23 row(s) affected" [23 linhas afetadas]. Isso indica quantas linhas foram retornadas pelo SELECT.
Você poderia escolher ver os resultados em uma grade, ao invés de em modo texto.

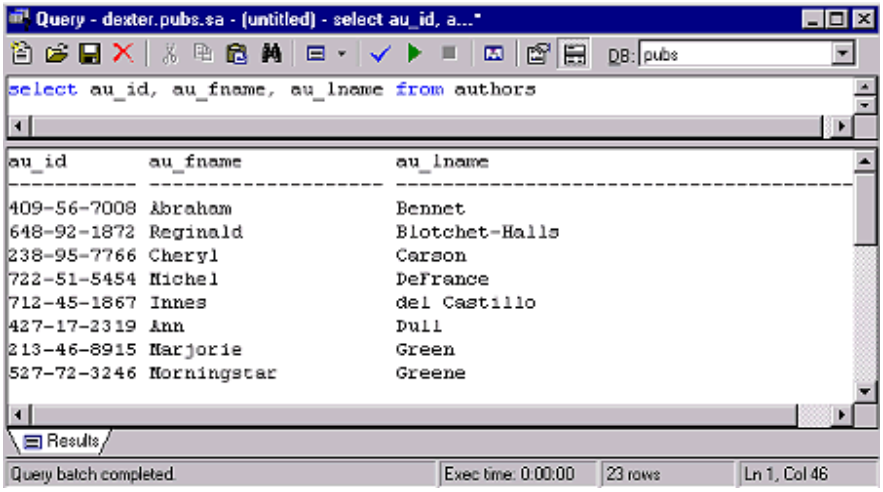
Para isso basta selecionar, clicando na seta à direita no ícone, a opção "Results in Grid" (Ctrl+D).

Caso você não queira selecionar na lista 'DB' o banco de dados a que a tabela que deseja procurar pertence, será necessário indicar no comando a qual banco de dados a tabela pertence, e o comando seria o seguinte:

```
select * from pubs..authors
```

Nessa tabela de exemplo, a coluna 'au_fname' é o primeiro nome do autor, 'au_lname' é o sobrenome [last name] e 'au_id' é o número de identificação. Agora suponhamos que você quer consultar apenas essas três primeiras colunas, omitindo a informação de telefone (phone) e endereço (address).
O '*' no comando acima especifica que todas as colunas da tabela são retornadas, mas você pode listar só as que são desejadas. Clique na página Query, e altere o comando anterior para o seguinte:

```
select au_id, au_fname, au_lname from authors
```

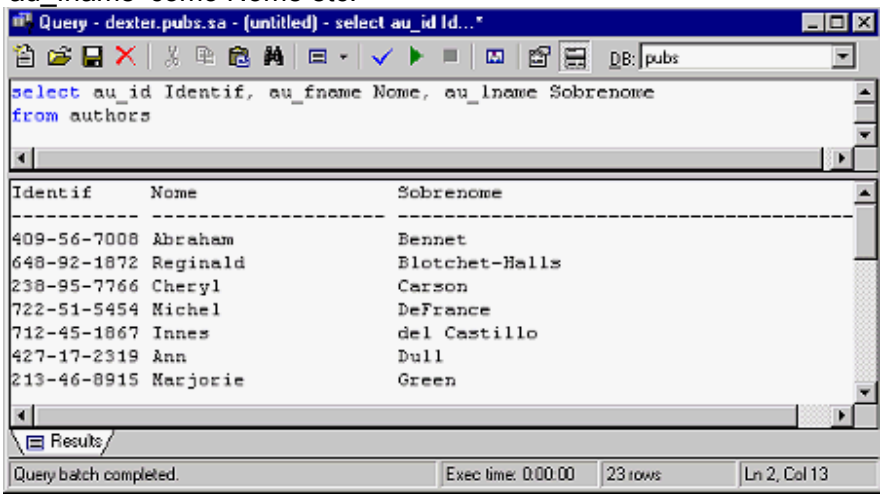


Execute o comando. Agora apenas as colunas 'au_id', 'au_fname' e 'au_lname' são retornadas, nessa ordem:

Note que a ordem das colunas não precisa ser a mesma ordem presente na definição da tabela.De fato, na maioria das aplicações bem construídas, a ordem das colunas na tabela não tem a menor importância.
Você também pode mudar o cabeçalho das colunas retornadas, criando um *alias* de coluna. Execute o seguinte comando:

```
select au_id Identif, au_fname Nome, au_lname Sobrenome
from authors
```

O resultado será o mesmo do comando anterior, mas a coluna 'au_id' aparece como Identif, 'au_fname' como Nome etc.



A palavra reservada **as** pode ser utilizada para indicar um alias, mas é opcional. Por exemplo:

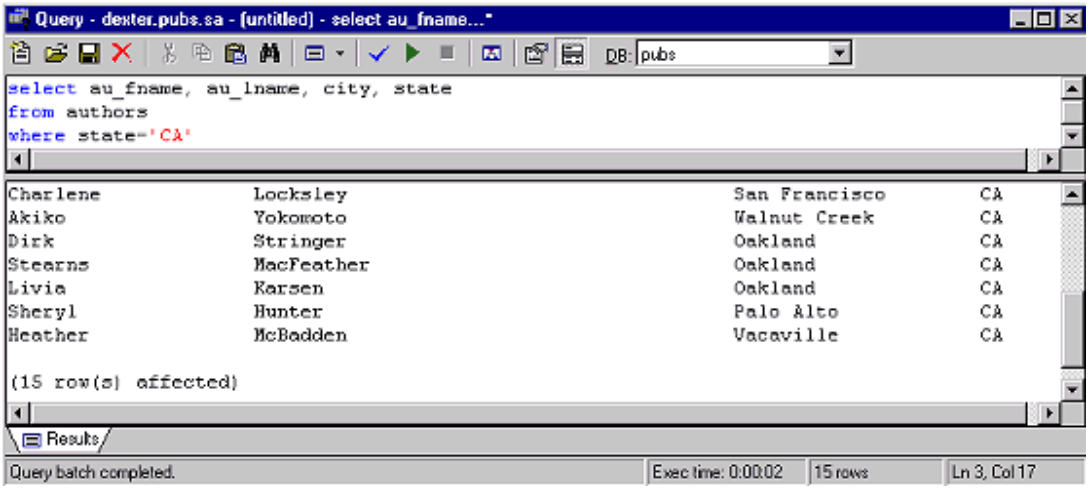
```
select au_id as Identif, au_fname as Nome, au_lname as Sobrenome
from authors
```

Usando Condições

Os comandos que já usamos não têm a cláusula WHERE. Nesse caso, todas as linhas da tabela são retornadas. Se o WHERE estiver presente, ele especifica uma condição que seleciona as linhas, e apenas as que satisfazem essa condição serão mostradas.
Por exemplo, se quisermos os autores que moram na Califórnia, podemos consultar as linhas cuja coluna 'state' (estado) tem o valor 'CA':

```
select au_fname, au_lname, city, state
from authors
where state='CA'
```

O resultado será:



Note que o resultado mostra apenas 15 linhas (15 rows affected) e não 23, que é o total da tabela. As linhas que aparecem são apenas as que satisfazem a consulta. Existem vários tipos de condições de pesquisa, como veremos.

Manipulando expressões

Um comando SELECT pode retornar nas colunas de resultado uma coluna da tabela, ou um valor calculado. Por exemplo, a tabela *titles* contém os títulos de livro (title) e os preços de cada um (price). Se quisermos ver como fica o preço de cada um após um aumento de 10%, pode ser feito o seguinte:

```
select price Preço , (price * 1.1) "Preço com 10% de
aumento", title from titles
```

Note que "Preço com 10% de aumento" é o nome do cabeçalho da expressão (price * 1.1), como o nome colocado possui espaços, foi necessário coloca-lo entre aspas.

Ou seja, a segunda coluna , cujo nome é "Preço com 10% de aumento " mostra o resultado de price * 1.1 para cada linha. Você pode também usar vários operadores em expressões com colunas numéricas: adição (+), subtração (-), multiplicação (*), divisão (/) e módulo (%). O módulo só pode ser usado com tipos inteiros e calcula o resto da divisão de dois números inteiros (Ex.: 13 % 4 = 1).

Funções matemáticas

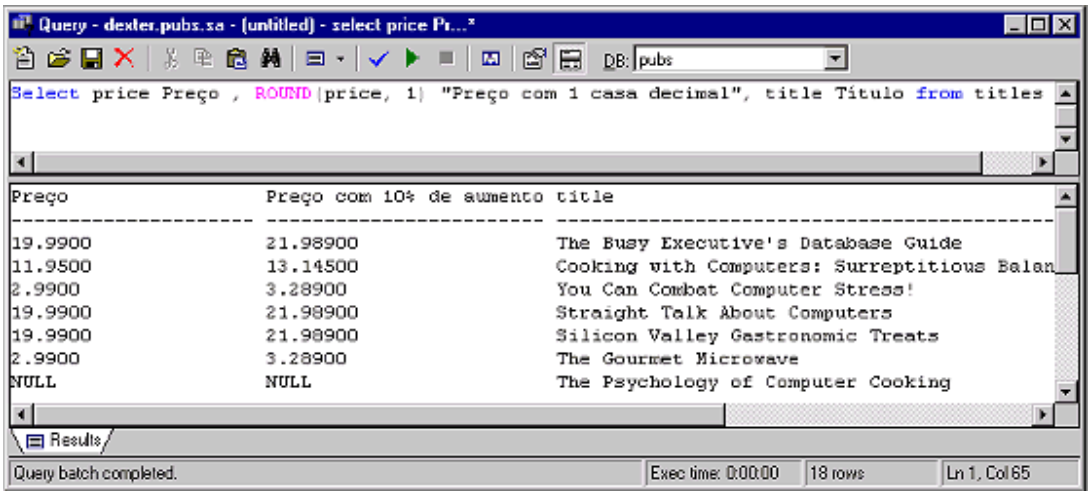
Além de operadores, você pode usar funções matemáticas do SQL Server, por exemplo:

ABS(*valor*)	retorna o valor absoluto (sem sinal) de um item.
POWER(*valor*,*p*)	retorna o *valor* elevado à potência *p*.
ROUND(*valor*,*n*)	arredonda o *valor* para *n* casas decimais.
SQRT (valor)	retorna a raiz quadrada do valor especificado.
PI	valor constante 3.141592563589793

Para outras funções, consulte o help do Transact-SQL em 'Math functions'.
Por exemplo, para arredondar o valor do preço de cada livro para duas casas decimais, pode ser feito o seguinte:

```
Select price Preço , ROUND(price, 1) "Preço com 1 casa
decimal", title Título from titles
```

O resultado será:

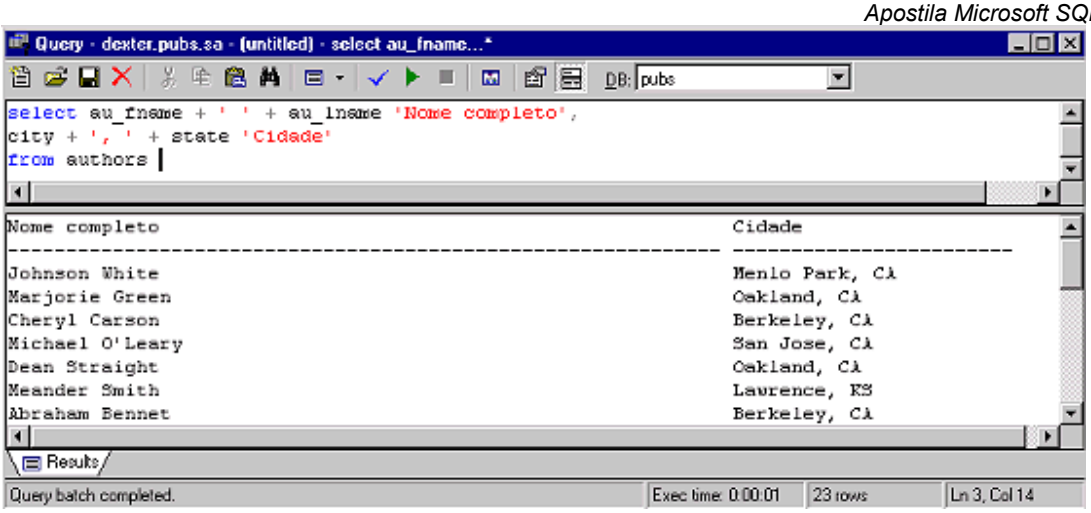


Funções de caracteres

Você pode usar funções para manipular dados do tipo caracter (*char* ou *varchar*), por exemplo, para pegar uma sub-string de uma seqüência de caracteres. E você pode usar o operador + para concatenar dois valores de tipo caracter.
Por exemplo, digite o seguinte comando:

```
select au_fname + ' ' + au_lname 'Nome completo',
city + ', ' + state 'Cidade'
from authors
```

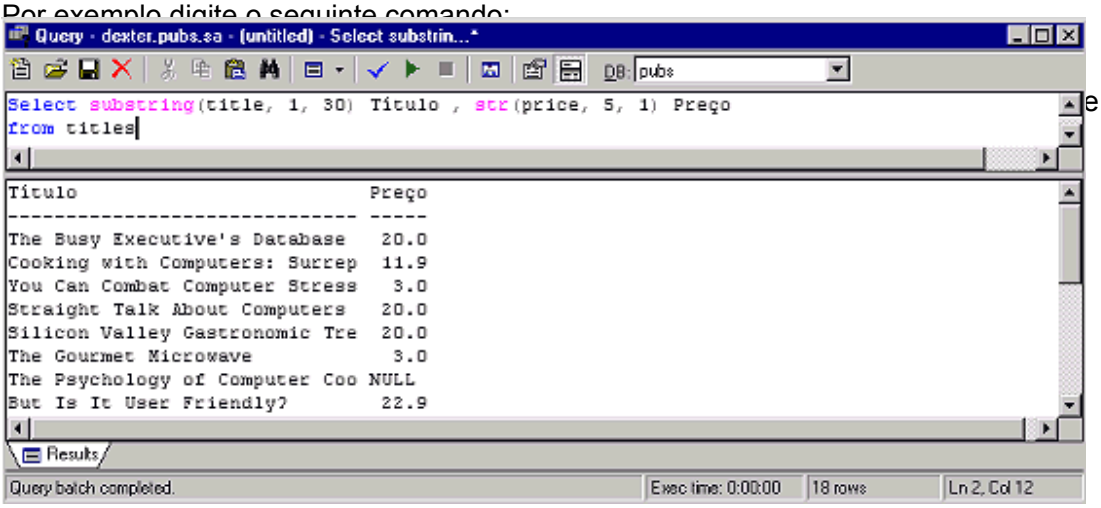
O resultado será:



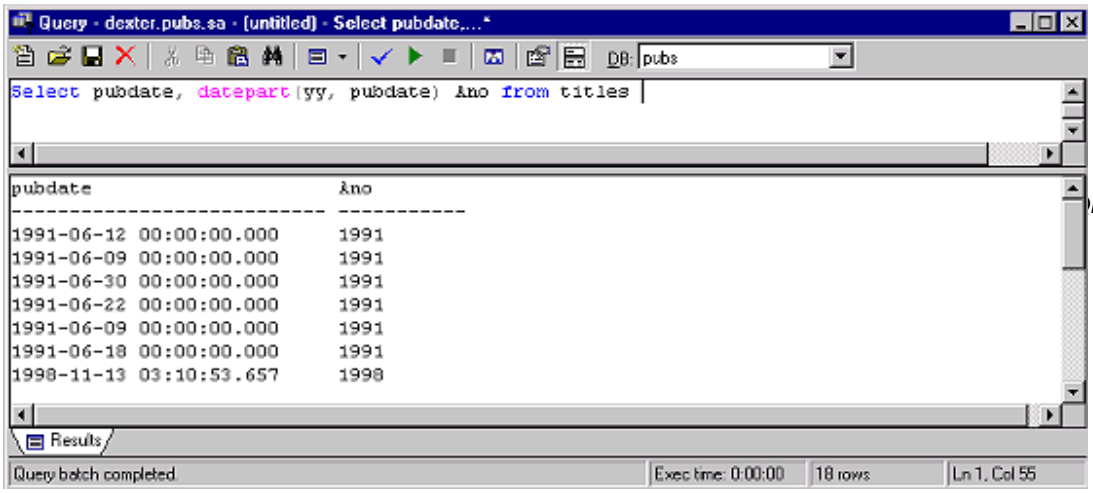
O que fizemos foi concatenar o primeiro nome e segundo nome do autor, inserindo um espaço no meio (au_fname + ' ' + au_lname), para gerar a coluna Nome completo e depois juntar o nome da cidade com o nome do estado, inserindo uma vírgula (city + ', '+state).
Existem várias funções de manipulação de strings que podem ser usadas para outras tarefas, por exemplo:

ASCII(*caractere*)	retorna o código ASCII de um caractere.
CHAR(*inteiro*)	retorna o caractere, dado o seu código ASCII
LOWER(*expr*)	converte para minúsculas
UPPER(*expr*)	converte para maiúsculas
LTRIM(*expr*)	retira espaços à esquerda
RTRIM(*expr*)	retira espaços à direita
REPLICATE(*expr*, *n*)	repete uma expressão *n* vezes
SUBSTRING(*expr,início,tamanho*)	extrai uma parte de uma string desde *início* e com *tamanho* caracteres
RIGHT(*expr,n*)	retorna *n* caracteres à direita da string
REVERSE(*expr*)	inverte uma string
CHARINDEX(*'caractere', expr*)	retorna a posição de um caractere dentro da string
SPACE(*n*)	retorna uma string com *n* espaços
STR(*número,n,d*)	converte um valor numérico para string, formatado com *n* caracteres na parte inteira (antes da vírgula) e *d* casas decimais depois da vírgula.
STUFF(*expr1,início,tamanho, expr2*)	substitui em *expr1*, os caracteres desde *início* até *tamanho* por *expr2*
DATALENGTH(*expr*)	retorna a quantidade de caracteres em *expr*

Por exemplo digite o seguinte comando:



e com no



Digite o seguinte comando :

```
Select Replicate('a', 10)
```

Com este comando a letra a foi mostrada 10 vezes.

Funções de Data/hora

O tipo *datetime*, como vimos, armazena datas e horas. Algumas funções trabalham com esse tipo de dados:

DATEADD(*parte,número,data*)	adiciona um certo número de dias (ou meses, anos etc.) à data
DATEDIFF(*parte,data1,data2*)	subtrai as duas datas (*data2 - data1*), retornando um resultado em dias, meses etc. dependendo de *datepart*
DATEPART(*parte,data*)	retorna a parte especificada da data
DATENAME(*parte,data*)	retorna o nome por extenso da parte especificada
GETDATE()	retorna a data e hora atuais

Nas funções acima, o argumento *parte,* especifica qual parte da data usar. Ele pode ser um dos seguintes valores:

yy	o ano
qq	o trimestre
mm	o mês
dy	o dia do ano (1-365)
dd	o dia do mês
wk	o número da semana (0-51)
dw	o dia da semana (domingo=1, segunda=2,...)
hh	a hora (0-23)
mi	os minutos
ss	os segundos
ms	os milisegundos

Por exemplo, digite o seguinte comando:

```
Select pubdate, datepart(yy, pubdate) Ano from titles
```

O resultado será:

O que fizemos foi mostrar o campo pubdate(Data de publicação do livro) e o de publicação do livro. Para mostrar o ano utilizamos a função de data datepart .

Conversão de dados

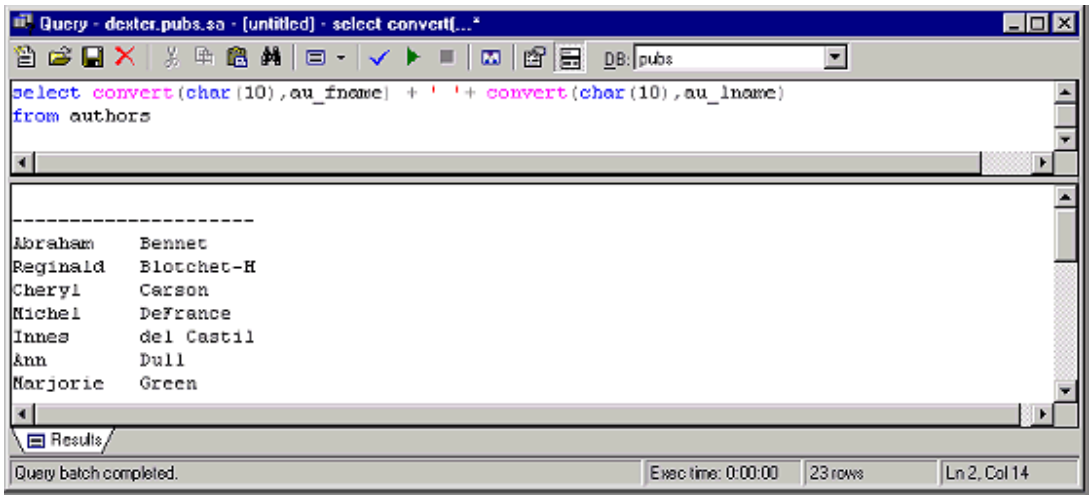
A função CONVERT permite converter de um tipo de dado para outro. A sua forma geral de uso é:

CONVERT(*tipo_de_dados*, *valor*)

Por exemplo:

```
select convert(char(10),au_fname) + ' '+ convert(char(10),au_lname)
from authors
```

Ao executar este comando o resultado será:



Com valores *datetime*, convert pode ter um parâmetro a mais, que especifica o formato de data a ser usado. Os formatos mais usados são 3 (padrão brasileiro dd/mm/aa), 103 (dd/mm/aaaa) e os padrões americanos 1 (mm/dd/yy) e 101 (mm/dd/yyyy). O default é 0, que mostra datas como:
Jan 01 1997 01:13:23 PM
Por exemplo, para ver a data de hoje em formato brasileiro, execute:

```
select convert(char,getdate(),103)
```

Para converter valores numéricos em char pode ser utilizado a função CONVERT, por exemplo:

```
select convert(char,pub_id) from titles
```

Condições de pesquisa

Como vimos, a cláusula WHERE permite selecionar quais as linhas da tabela a serem incluídas no resultado. Existem várias formas de montar uma cláusula WHERE, usando um dos seguintes elementos:
Operadores de comparação:

=	igual a
>	maior que
<	menor que
>= ou !<	maior ou igual (não menor)
<= ou !>	menor ou igual (não maior)
<> ou !=	diferente

Faixas:	BETWEEN *valor-ini* AND *valorl-fin*
Listas:	IN (*lista*)
Casamento de padrões:	LIKE *padrão*
Valores nulos:	IS NULL, IS NOT NULL
Combinação de condições:	AND, OR, NOT

Usando operadores

As condições mais simples são formadas usando operadores de comparação, como vimos no exemplo anterior:

```
select au_lname, city from authors
where state = 'CA'
```

Note que constantes do tipo *char* (ou *varchar*), bem como datas, devem ser colocadas entre apóstrofos (').
Se quisermos fazer o contrário, isto é, procurar os autores que NÃO são da Califórnia, podemos fazer:

```
select au_lname, city from authors
where state <> 'CA'
```

Note que *diferente* também pode ser representado por !=.

Usando faixas

Na tabela *titles*, para cada livro, está guardada a sua data de publicação na coluna 'pubdate'. Se quisermos saber quais os livros publicados no ano de 1991, podemos fazer a consulta:

```
select pubdate, title
from titles
where pubdate between '1/1/91' and '12/31/91'
```

Onde BETWEEN...AND... seleciona os valores de 'pubdate' que estão dentro de uma determinada faixa. Para fazer o contrário, bastaria usar NOT BETWEEN (mas nesse caso não é tão eficiente a consulta).

Usando listas

Você pode selecionar valores de acordo com uma lista. Se o valor pertence à lista, a linha será incluída no resultado. Por exemplo:

```
select au_lname, city, state
from authors
where state in ('UT','CA')
```

O resultado contém as linhas onde 'state' tem um dos valores 'UT' ou 'CA'. Equivale ao mesmo que usar uma condição composta:

```
where state = 'UT' OR state = 'CA'
```

Para retornar os valores que *não* estão na lista, pode-se usar NOT IN.

Casamento de padrões

O operador LIKE [como] faz casamento de padrões. Um *padrão* é uma string contendo caracteres que podem ser combinados com parte de outra string.
Por exemplo, o caractere % em um padrão representa qualquer quantidade de caracteres. Por exemplo, para obter todos os autores cujo (primeiro) nome começa com A, use:

```
select au_fname, au_lname from authors
where au_fname like 'A%'
```

Para obter todos os nomes que contém as letras 'en' no meio (ou no início ou no fim), use:

```
select au_fname, au_lname from authors
where au_fname like '%en%'
```

Outro caractere para usar em padrões é o sublinhado (_). Ele combina com um único caractere. Por exemplo, se nos seus dados existem pessoas com nome 'Sousa' ou 'Souza', você pode usar: LIKE '%sou_a%'.
Finalmente, é possível usar os colchetes para combinar com uma determinada faixa de caracteres. Por exemplo, LIKE '[CK]%' encontra os nomes que iniciam com C ou K e LIKE '[A-E]%' os que começam com as letras de A até E. Já LIKE '[^V]%' encontra os nomes que *não* começam com V (o caractere ^ indica não).

Note que as comparações feitas com LIKE dependem da ordem de classificação [*sort order*] escolhida durante a instalação do SQL Server. Se foi usada a ordem "accent-insensitive", como foi recomendado, ele consegue procurar ignorando acentos. Por exemplo, LIKE 'camara' vai encontrar também 'Câmara'.

Procurando valores nulos

O valor NULL no SQL Server indica "não informado" ou "desconhecido". Ele é inserido numa coluna quando aquele valor não é conhecido ou não se aplica. Praticamente qualquer coluna pode ter valores NULL, exceto se tiver sido declarada como NOT NULL na criação da tabela. NULL não é tratado como outros valores. Especialmente, qualquer operação com NULL tem como resultado NULL e qualquer comparação com NULL tem resultado FALSO. Por exemplo, veja as duas consultas a seguir:

```
select price, title from titles
where price = 19.99

select price, title from titles
where price <> 19.99
```

Algumas linhas da tabela 'title' têm 'price'=NULL. Essas linhas não vão aparecer nem na primeira nem na segunda consulta, porque NULL=19.99 é falso e NULL<>19.99 também é falso. Para ver essas linhas com valores nulos, use:

```
select price, title from titles
where price is NULL
```

Para ver as linhas onde o valor está preenchido (não NULL), use:

```
select price, title from titles
where price is NOT NULL
```

Usar 'price = NULL' funciona no SQL Server, mas não é compatível com o padrão ANSI.

Juntando várias condições

Você pode fazer condições compostas com AND, OR ou NOT.
O operador AND (E) liga duas condições e retorna verdadeiro apenas se *ambas* são verdadeiras e falso se pelo menos uma delas é falsa. Já OR (OU) retorna verdadeiro se *pelo menos uma* delas for verdadeira e falso se ambas forem falsas. O operador NOT (NÃO) inverte uma condição. Por exemplo:

```
select title, pub_id, price
from titles
where (title like 'T%' OR pub_id = '0877')
AND (price > $16.00)
```

Isso retorna os livros onde:
Ambas as condições 1 e 2 são verdadeiras:
1-*Uma das* seguintes é verdadeira
o título ('title') começa com T,
OU
o código da editora ('pub_id') é '0877'
E

2- o preço ('price') é maior que 16.00
Os parênteses indicam a precedência das condições. Na falta de parênteses, o operador NOT, se presente, é aplicado primeiro, depois as condições com AND são agrupadas, depois as condições com OR são agrupadas. Você pode usar parênteses, mesmo se não necessários, para tornar a expressão mais legível.

Outros recursos

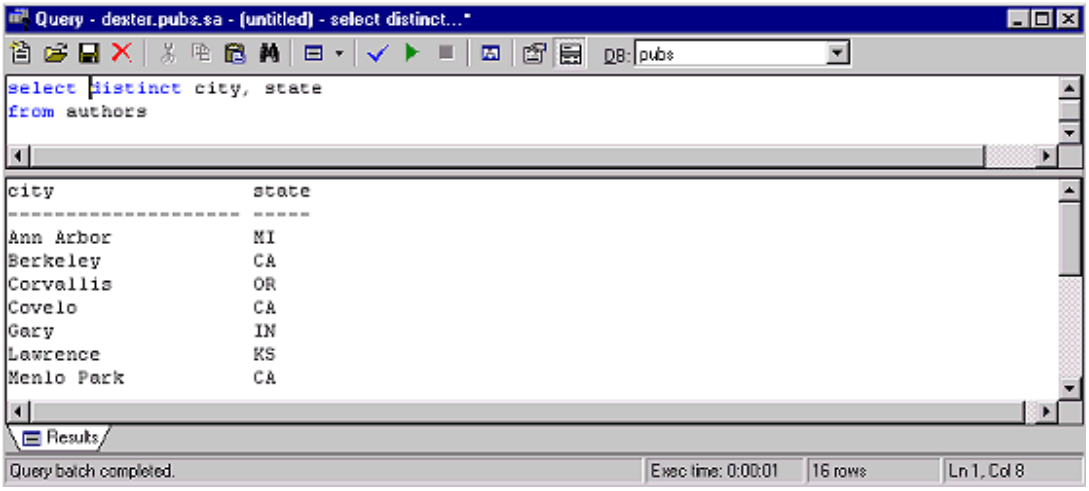
No comando SELECT, a cláusula DISTINCT elimina valores duplicados no resultado e ordena a lista de resultados. Para apenas ordenar por uma ou mais colunas, use ORDER BY. É possível também combinar o resultado de dois comandos SELECT em um único conjunto com o operador UNION.

Eliminando valores duplicados

Se você quiser saber quais as cidades e estados nas quais mora algum autor, pode usar a seguinte consulta:

```
select city, state
from authors
```

Mas note que algumas cidades, como Oakland, CA, aparecem várias vezes. Para eliminar duplicações, use a cláusula DISTINCT. O SQL leva em conta as duas colunas em conjunto, para remover duplicatas.




```
select distinct city, state
from authors
```

O resultado será:

Note que o resultado terá apenas 16 linhas. O SQL Server ordena os dados implicitamente, para poder eliminar as duplicatas.

Ordenando resultados

Para ver o resultado numa ordem particular, use a cláusula ORDER BY. Se estiver presente, deve ser a última cláusula do comando SELECT. Por exemplo, para ver os livros em ordem de preço:

```
select title, type, price
from titles
order by price
```

Você pode indicar após o nome da coluna, se a ordem é ascendente ou descendente, por exemplo:

```
select title, type, price
from titles
order by type asc, price desc
```

Se nem ASC nem DESC estiverem presentes, o default é ASC.
Em vez de colocar o nome da coluna, você pode usar o número relativo (1,2,...). A mesma consulta anterior poderia ser:

```
select title, type, price
from titles
order by 2 asc, 3 desc
```

União de conjuntos

O comando SELECT retorna um *conjunto* de linhas, e permite também operações com a noção matemática de conjuntos. Por exemplo, o resultado de dois comandos SELECT pode ser combinados com o operador UNION. Os dois comandos podem até mesmo trazer dados de tabelas diferentes, desde que com o mesmo número de colunas, e tipos de dados compatíveis para cada coluna correspondente de um com o outro.
Por exemplo, no banco de dados *pubs*, a tabela *authors* contém informação sobre cada autor, o que inclui a cidade e estado onde ele mora (colunas *city* e *state*). A tabela *publishers* contém informação sobre as editoras e suas cidades e estados. Para sabermos o conjunto de todas as cidades onde existem autores ou editoras, pode ser feita uma união dos dois conjuntos, com:

```
select city, state from authors
union
select city, state from publishers
```

Note que na união de dois conjuntos, os elementos repetidos são eliminados, como quando se usa o DISTINCT. O resultado também aparece em ordem crescente, pois o SQL Server ordena os resultados antes de eliminar repetições.
Se você quer ordenar de modo diferente os resultados da união, usando ORDER BY, essa cláusula só pode aparecer no segundo comando SELECT, ou seja, no final dos comandos, por exemplo:

```
select city, state from authors
union
select city, state from publishers
order by state
```

Alteração de Dados

Inserindo Linhas

Atualizando Linhas

Excluindo Linhas

Objetivos:
- Aprender a inserir , atualizar e excluir linhas.

Inserindo linhas

O comando INSERT insere linhas em uma tabela. A forma mais simples do comando INSERT insere somente uma linha , dados os valores.

Sintaxe

```
INSERT [INTO] nome_tabela (colunas)
VALUE (valores)
```

Onde:
nome_tabela é o nome da tabela que deseja incluir os dados.
colunas é o nome das colunas da tabela que deseja acrescentar os dados.
valores é o conteúdo de cada coluna.
Exemplos:
Vamos usar a tabela 'funcionario', do banco de dados Exemplo, criada anteriormente, para inserir linhas. Na janela SQL Query Tool, selecione em "BD" o banco de dados Exemplo. Digite o seguinte:

```
insert into Funcionario
values (1, 'Primeiro Funcionário', 2, 122, 234.23,
'01/01/1998', '01/01/1998', 'M')
```

Nesse caso, são informados os valores de todas as colunas da tabela, na ordem em que elas foram definidas na tabela. Mas é possível também inserir dados parciais de apenas algumas colunas. Para testar, digite os seguintes comandos e depois execute:

```
insert into Funcionario (CodFuncionario, Nome, CodDepartamento, Sexo,
Salario, DataAdmissao)
values (2, 'Segundo Funcionário', 1, 'F', 4360.00, '01/01/1996')

insert into Funcionario (CodFuncionario, Nome, CodDepartamento, Sexo,
Salario, DataAdmissao, Ramal)
values (3, 'Terceiro Funcionário', 1, 'F', 1500.00, '12/30/1995', 122)

insert into Funcionario (CodFuncionario, Nome, CodDepartamento, Sexo,
Salario, DataAdmissao)
values (4, 'Quarto Funcionário', 1, 'M', 1500.34, '10/30/1996')

insert into Funcionario (CodFuncionario, Nome, CodDepartamento, Sexo,
Salario, DataAdmissao)
values (5, 'Quinto Funcionário', 3, 'M', 500.34, '07/30/1997')

insert into Funcionario (CodFuncionario, Nome, CodDepartamento, Sexo,
Salario, DataAdmissao)
values (6, 'Sexto Funcionário', 3, 'F', 1000.34, '08/30/1995')

insert into Funcionario (CodFuncionario, Nome, CodDepartamento, Sexo,
Salario, DataAdmissao)
values (7, 'Sétimo Funcionário', 3, 'F', 900.34, '01/01/1997')
```

Nesse caso, os nomes das colunas que serão inseridas são especificados entre parênteses após o nome da tabela. A ordem não precisa ser a mesma das colunas na tabela. Mas a ordem dos valores em VALUES corresponde à ordem dos nomes de colunas informados.
Nesse caso, se uma coluna é omitida da lista, o SQL Server faz o seguinte:
• Se a coluna tem um valor default , o valor default é inserido.
• Caso contrário, se a coluna permite valores NULL, será inserido um NULL.
• Caso a coluna não tenha default e tenha sido criada como NOT NULL, o SQL Server gera uma mensagem de erro e cancela a execução do comando.
Digite agora 'SELECT * from Funcionario'. Você verá que os funcionários 2 , 3 e 4 a data de cadastro é a data de hoje, porque o default para essa coluna é a função GETDATE(). Em outros casos, o valor da coluna ficou NULL quando não informado.
A palavra reservada DEFAULT insere o valor default da coluna. Como exemplo execute o seguinte comando:

```
insert into Funcionario
values (8, 'Oitavo Funcionário', 2, 122, 600.23, '01/01/1998', DEFAULT, 'M')
```

Digite agora 'Select * from funcionario where codfuncionario = 8'.Será mostrado os dados do funcionário cujo código é igual a 8. O conteúdo da data do cadastro é a data de hoje, isto ocorreu devido ao seu valor default.
Não acrescente nenhuma linha a mais na tabela de 'funcionario', porque ela será usada posteriormente nos nossos exemplos. Caso acrescente o resultado dos exemplos que iremos utilizar não irá coincidir.
Vamos inserir alguns dados na tabela 'departamento' do banco de dados Exemplo.Esta tabela possui três colunas. A coluna CodDeptSuperior indica o código do departamento que o departamento que esta sendo cadastrado é subordinado.
Execute os seguintes comandos:

```
insert into Departamento Values (1, 'Diretoria', 0)

insert into Departamento Values (2, 'Departamento Administrativo', 1)

insert into Departamento Values (3, 'Departamento Pessoal', 1)
```

Não acrescente nenhuma linha a mais na tabela de 'departamento', porque ela será usada posteriormente nos nossos exemplos. Caso acrescente o resultado dos exemplos que iremos utilizar não irá coincidir.
Acrescente dados para a tabela de clientes. Observe que as colunas DataCadastro e País possuem valores Default , as colunas CPF, Cidade, Estado pode conter o valor null.

Usando INSERT com SELECT

Você também pode inserir o resultado de uma consulta SELECT dentro de uma tabela. Para testar, crie uma nova tabela no banco de dados Exemplo, usando o Enterprise Manager (ou com o comando CREATE TABLE), com o nome de CopiaCliente. A tabela deverá ter as seguintes colunas:

Nome	**Tipo**
Codigo	int
Nome	varchar(50)

Para copiar as linhas 2 e 4 da tabela Cliente, use o seguinte comando:

```
insert into CopiaCliente
   select CodCliente, Nome
   from Cliente
   where CodCliente in (2,4)
```

Cada linha retornada pelo SELECT interno será inserida na tabela CopiaCliente. Esse comando é muito útil para copiar dados entre tabelas semelhantes. Note que nesse caso, as regras que vimos anteriormente ainda se aplicam, a cada linha que o comando está tentando inserir. As colunas da tabela de destino e os valores de resultado do SELECT devem ser compatíveis, ou seja, devem ter o mesmo tipo de dados ou tipos compatíveis e devem estar na mesma ordem (mas os nomes não precisam ser os mesmos, como no caso de 'CodCliente' e 'Codigo').
Se as duas tabelas fossem idênticas, poderia ser usado * no select em vez de uma lista de colunas. Se a tabela CopiaCliente tivesse colunas a mais, além de CodCliente e Nome, teria de ser especificada a lista de colunas a ser inseridos, como vimos anteriormente.

Excluindo linhas

O comando DELETE exclui permanentemente uma ou mais linhas de uma tabela, baseado em alguma condição.
Sintaxe

```
DELETE FROM nome_tabela WHERE condicao
```

Onde:
nome_tabela é o nome da tabela que deseja excluir os dados.
condicao é condição para selecionar as dados que deseja excluir.
Por exemplo, para excluir o cliente nº 2 (Codigo =2) da tabela CopiaCliente, execute o seguinte comando no banco de dados Exemplo:

```
delete from CopiaCliente
   where Codigo = 2
```

Note que a exclusão não pode ser desfeita.

Usando sub-consultas

Assim como UPDATE, o comando DELETE também pode usar sub-consultas para excluir linhas baseado nos dados de outra tabela.
Para testar, copie novamente as linhas de cliente para CopiaCliente e depois insira duas novas linhas em CopiaCliente, como abaixo:

```
insert into CopiaCliente
  select CodCliente, Nome from Cliente
insert into CopiaCliente (5, 'Cliente Cinco')
insert into CopiaCliente (6, 'Cliente Seis')
```

Agora vamos apagar de CopiaCliente apenas as linhas que existem na tabela Cliente. Para isso, use o comando abaixo:

```
delete from CopiaCliente
where Codigo in
   (select CodCliente from Cliente)
```

Limpando uma tabela

Para excluir todas as linhas de uma tabela, existem duas opções. Uma é usar um comando DELETE sem condição WHERE:

```
delete from CopiaCliente
```

Outra opção é o comando TRUNCATE TABLE, que quase sempre é mais rápido que o DELETE, especialmente em tabelas grandes:

```
truncate table CopiaCliente
```

Mas TRUNCATE TABLE não salva informações no log de transações, o que tem algumas conseqüências com relação a backups, como veremos.

8 - Consultas Avançadas

Dados de Resumo

Junções de Tabelas

Sub-consultas

Objetivos:
- Aprender a gerar dados de resumo com funções agregadas e GROUP BY;
- Aprender a consultar dados a partir de duas ou mais tabelas.

Dados de resumo

Além da sintaxe básica do SELECT que já vimos, alguns elementos a mais podem ser incluídos, as cláusulas GROUP BY e HAVING:

Sintaxe

SELECT *lista_de_colunas*
FROM *lista_de_tabelas*
WHERE *condições*
GROUP BY *lista_de_expressões*
HAVING *condições*

Como já vimos, o WHERE (se presente) separa as linhas que satisfazem as condições iniciais. A cláusula GROUP BY organiza as linhas de resultado em grupos de acordo com os valores das expressões informadas. A cláusula HAVING (opcional) seleciona os grupos de acordo com os resultados. O resultado do SELECT...GROUP BY... tem uma linha para cada grupo, que pode conter *valores de resumo* (somatório, média, contagem etc.) calculados dentro do grupo. Quando o GROUP BY está presente, é possível usar *funções agregadas*, que calculam valores baseado nas linhas de um grupo e geram valores de resumo.

Exemplo

Para saber quantas linhas existem na tabela *'funcionario'* do banco de dados Exemplo, pode-se usar a função agregada COUNT(*):

```
select count(*)
from funcionario
```

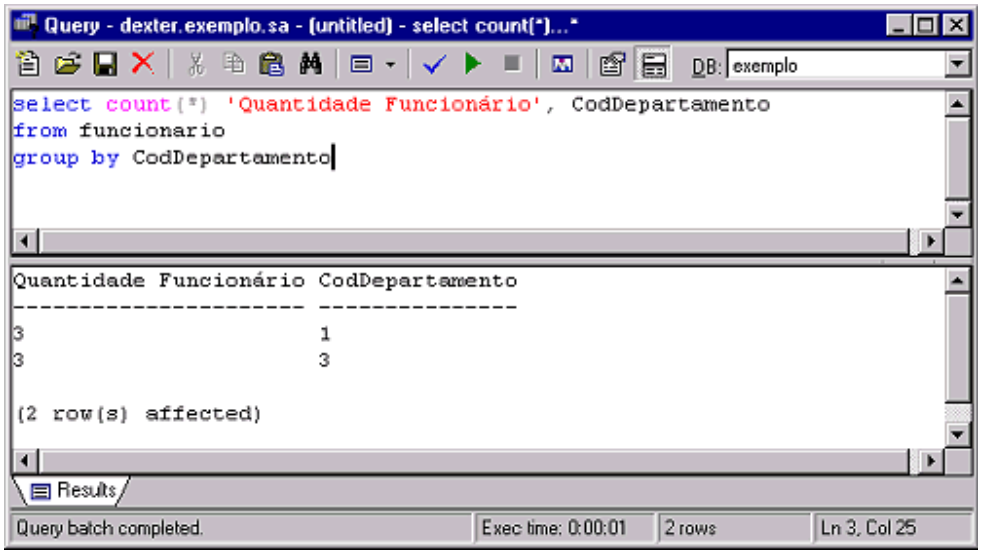
O resultado será 8. Quando a cláusula GROUP BY é omitida, como nesse exemplo, todas as linhas da tabela são agrupadas para formar uma linha de resultado. Você pode também acrescentar uma condição:

```
select count(*)
from funcionario
where ramal is not null
```

Neste caso o resultado é 3. Estamos contando quantos funcionarios tem a coluna Ramal diferente de null. Quando se utiliza o GROUP BY com o nome de uma coluna, os resultados são agregados por essa coluna. Por exemplo, para saber quantos funcionarios existem por departamento:

```
select count(*) 'Quantidade Funcionário', CodDepartamento
from funcionario
group by CodDepartamento
```

O resultado será:



Ou seja, todos os três funcionários cujo código do departamento é 1 foram agrupados para gerar uma só linha de resultado. Depois todos os funcionários com coddepartamento = 3 (caso houvesse algum funcionário cadastrado com código departamento = 2, viria logo após o de código departamento = 1) e assim por diante. No resultado é mostrada a coluna 'coddepartamento' (que foi usada no GROUP BY para definir os grupos) e o resultado da função COUNT(*), que é a contagem de elementos de cada grupo.

Funções agregadas

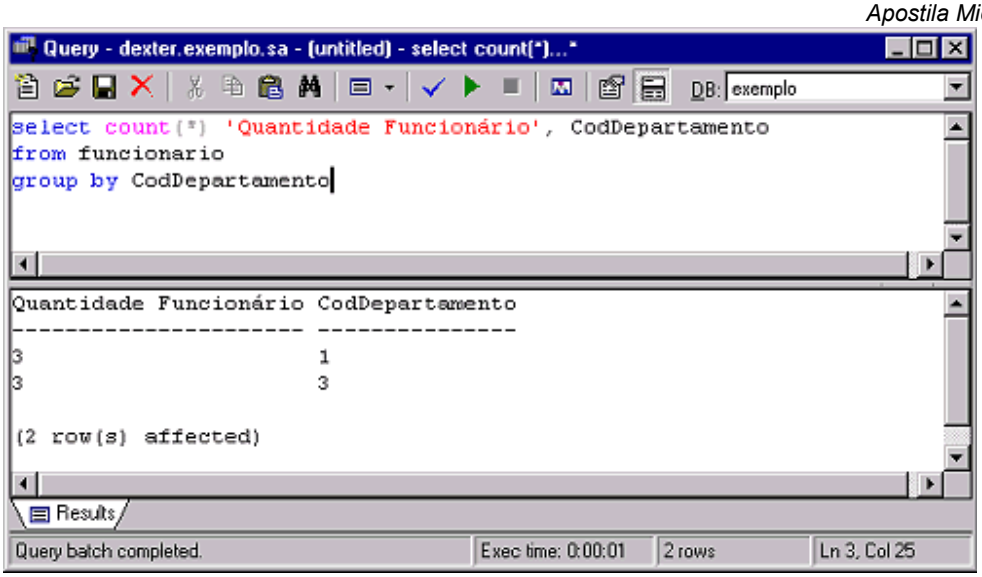
Além da função COUNT, existem outras funções agregadas que podem ser usadas para fazer operações sobre os elementos do grupo:

AVG(*expr*)	Calcula o valor médio da expressão *expr* dentro do grupo. A expressão pode ser um nome de coluna ou calculada a partir de colunas e/ou constantes. (Por exemplo, `AVG (salario*1.1)`)
COUNT(*expr*)	Conta quantos valores existem da expressão dada dentro do grupo (se *expr* for NULL para uma linha, a linha não é incluída na contagem).
COUNT(*)	Conta quantas linhas existem dentro do grupo.
MAX(*expr*)	Retorna o máximo valor de *expr* dentro do grupo.
MIN(*expr*)	Retorna o mínimo valor de *expr* dentro do grupo.
SUM(*expr*)	Retorna o somatório da expressão dentro do grupo.

As funções AVG e SUM podem ser usadas apenas com dados numéricos. As outras podem ser usadas com qualquer tipo de coluna. As funções SUM, AVG e COUNT(*expr*) permitem especificar também o operador DISTINCT, que indica para considerar apenas os valores distintos. Por exemplo, a tabela 'funcionarios' do banco de dados Exemplo contém informação sobre qual departamento este funcionário trabalha. Para saber quais os departamentos que têm ao menos um funcionario , execute o seguinte comando:

```
select count(distinct coddepartamento)
from funcionario
```

O resultado é 3, o número de valores *distintos* de 'departamentos'. Sem usar DISTINCT, o resultado seria 8 (o número de linhas da tabela 'funcionario').



Outro exemplo usando a tabela 'funcionario': para saber o total de salários pagos por cada departamento, pode-se usar GROUP BY e a função SUM calculando o somatório da coluna 'Salario':

```
select coddepartamento Departamento, Sum(Salario) 'Total Salário'
from funcionario
group by coddepartamento
```

O resultado será:

Detalhes do GROUP BY

A cláusula GROUP BY agrupa valores baseado em uma ou mais colunas. No último SELECT acima, group by coddepartamento significa que todas as linhas que têm o mesmo valor da coluna 'coddepartamento' serão agrupadas em uma só. Uma função agregada, como SUM, COUNT, AVG calcula valores sobre todos os elementos do grupo. Uma linha de resumo é gerada para o grupo, contendo o valor representante do grupo, 'coddepartamento' e o resultado de SUM(Salario).

Note que as colunas de resultado da cláusula SELECT (a lista de colunas após o SELECT) podem ser apenas:

- Uma coluna presente na lista do GROUP BY

OU

- Um valor gerado por uma função agregada

Outras colunas não podem ser incluídas no resultado, porque teriam valores diferentes para cada linha do grupo.

Usando a cláusula HAVING

Após feito o agrupamento, pode-se usar a cláusula HAVING para selecionar quais os grupos a serem incluídos no resultado. Por exemplo, para selecionar os departamentos que pagam mais que 1000.00, pode-se fazer:

```
select coddepartamento Departamento, sum(Salario) 'Salário Total'
from funcionario
group by coddepartamento
having sum(Salario) > 1000.00
```

Note que as cláusulas WHERE e HAVING são semelhantes. Mas WHERE seleciona as linhas da tabela que irão participar da geração do resultado. Essas linhas serão agrupadas e *depois*

HAVING é aplicado ao resultado de cada grupo, para saber *quais grupos* vão aparecer no resultado. Nas condições usadas por HAVING só podem aparecer valores que sejam os mesmos em todos os elementos do grupo.

Junções de tabelas

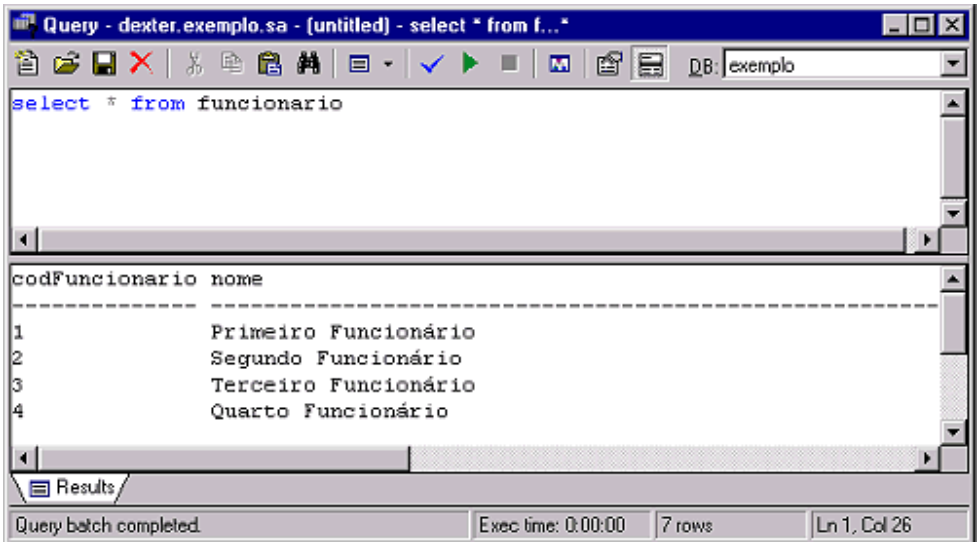
Um comando SELECT também pode fazer uma consulta que traz dados de duas ou mais tabelas. Esse é um processo chamado de *junção* [join]. As tabelas têm uma coluna em comum que é usado para fazer as junções.
Antes de executar os exemplos mostrados abaixo, vamos inserir um departamento na tabela 'departamento' que neste caso não terá nenhum funcionário relacionado. Execute o seguinte comando:

```
insert into departamento values (4, 'Contabilidade', 2)
```

Por exemplo, no banco de dados *Exemplo*, a tabela 'departamento' contém dados de departamentos. Cada departamento tem um número de identificação único, 'CodDepartamento'. Na tabela 'funcionario' estão os dados dos funcionários. Para identificar o departamento do funcionário, a tabela 'funcionario' tem também uma coluna 'CodDepartamento' que pode ser usada para procurar na tabela 'departamento'. Por exemplo, digite:

```
select * from funcionario
```

O resultado será:

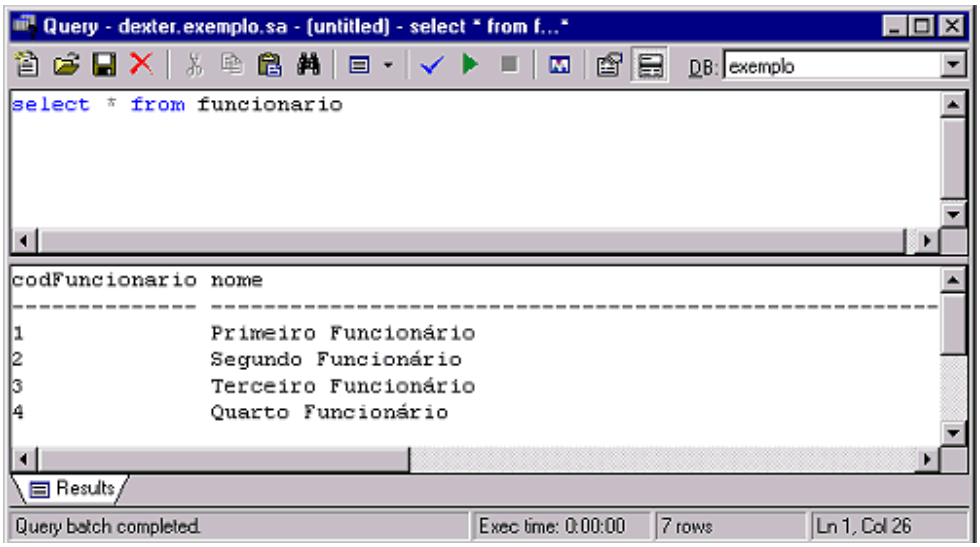


...e veja que na primeira linha da tabela, o nome do funcionário é 'Primeiro Funcionário' e o valor da coluna 'coddepartamento' é 2. Agora digite:

```
select * from departamento
```

Depois de executar, note que na segunda linha o nome do departamento , cujo código é igual a 2 , é 'Departamento Administrativo'.
Se quisermos ver uma listagem mostrando os funcionários e seus respectivos departamentos, basta fazer uma junção das duas tabelas. Uma das formas de fazer isso é:

```
select departamento.nome 'Departamento', funcionario.nome
```




```
from funcionario, departamento
where funcionario.coddepartamento = departamento.coddepartamento
```

O resultado será:

Neste exemplo como as duas tabelas tem o mesmo nome de coluna é necessário qualificar o nome com o nome da tabela, como "funcionario.nome", "departamento.nome". Em geral, é recomendável sempre qualificar os nomes para maior clareza.

Sintaxe da Junção

O SQL Server aceita duas sintaxes diferentes para junção de tabelas. Uma delas, mostrada acima, é específica ao SQL Server e, às vezes, um pouco mais simples de utilizar.
Na sintaxe do SQL Server, na lista do FROM as duas (ou mais) tabelas são especificadas, separadas por vírgulas. Na cláusula WHERE deve haver uma condição ligando as duas, a *condição de junção* [join condition]. Na lista de colunas do SELECT podem ser incluídos colunas de qualquer uma das tabelas.
No exemplo acima foram especificadas as colunas 'departamento.nome' (o nome da departamento, que vem da tabela 'departamento') e 'funcionario.nome' (nome do funcionario, que vem da tabela 'funcionario').
A outra forma de sintaxe que pode ser usada é a sintaxe do padrão ANSI SQL. O exemplo anterior, com a sintaxe ANSI, ficaria:

```
select departamento.nome, funcionario.nome
from funcionario inner join departamento
on funcionario.coddepartamento = departamento.coddepartamento
```

Nessa sintaxe, o *tipo de junção* entre as tabelas deve ser especificado entre elas (veremos os diferentes tipos abaixo) e a condição de junção é especificada com a palavra ON.

Junção interior

O exemplo acima é uma *junção interior* de tabelas [inner join]. Esse tipo de junção conecta as duas tabelas e retorna apenas as linhas que satisfazem a condição de junção. No exemplo, isso significa que, se existirem funcionários para os quais não há departamento relacionado eles não serão incluídos no resultado. Igualmente, se existirem departamentos que não têm empregados(como 'Contabilidade') , eles não aparecem no resultado.

Uma junção interior é chamada de *equijoin* quando as colunas são comparadas usando o =, e as duas colunas aparecem no resultado, mostrando dados redundantes, já que elas têm o mesmo valor. Uma junção interior é chamada *junção natural* quando a coluna usada para junção aparece apenas uma vez no resultado, vinda de uma ou outra tabela.
Na sintaxe ANSI, junções interiores são indicadas da forma:

```
tabela1 INNER JOIN tabela2 ON condição_de_junção
```

Junção cruzada ou irrestrita

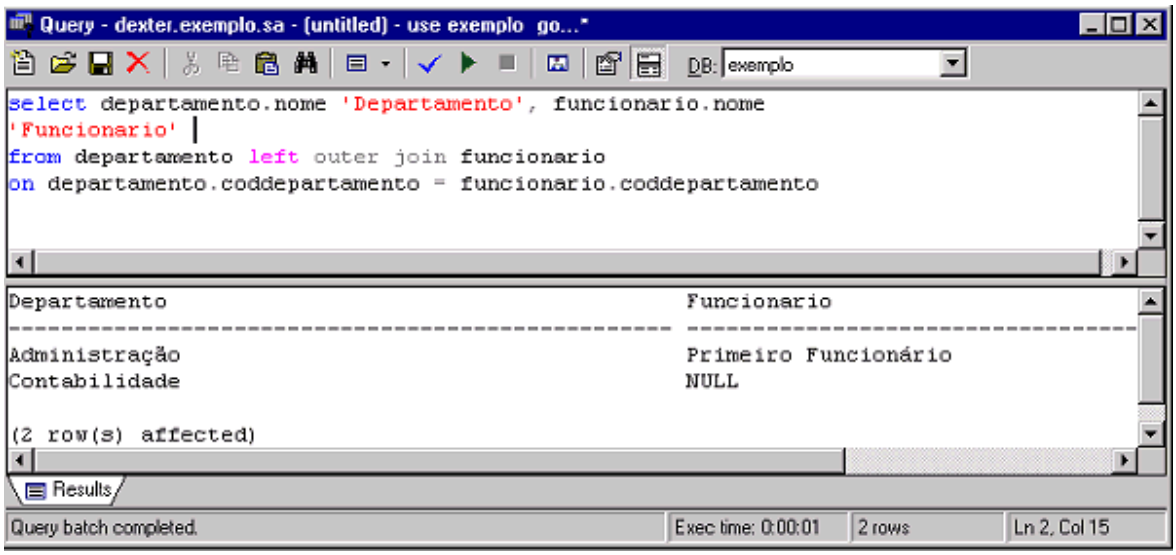
Uma *junção cruzada* [cross join] de tabelas, também chamada *junção irrestrita* de duas tabelas gera um resultado formado por *todas* as combinações possíveis de uma linha da primeira tabela com uma linha da segunda. Não existe uma condição de junção. Esse resultado é chamado *produto cartesiano*das duas tabelas. Na sintaxe ANSI, junções cruzadas são indicadas com CROSS JOIN, por exemplo:

```
select departamento.nome, funcionario.coddepartamento
from funcionario cross join departamento
```

Nesse caso, como a tabela 'funcionario' tem 8 linhas e a tabela 'departamento' tem 4 linhas , o resultado final será 32 linhas que é 8 * 4 = 32 linhas, formadas por todas as combinações possíveis de funcionarios e departamento.
Na sintaxe do SQL Server, basta especificar a lista de tabelas, sem usar uma condição de junção:

```
select departamento.nome, funcionario.nome, salario
from funcionario, departamento
```

Esse é exatamente um dos problemas com a sintaxe informal: se você incluir mais de uma tabela mas não colocar uma condição de junção(é fácil esquecer uma condição quando existem várias outras envolvidas), o banco de dados vai simplesmente fazer um produto cartesiano sem nenhum aviso.



Junções cruzadas raramente são usadas, mas é importante saber como evitar usá-las. Para isso, sempre crie uma junção interior ou exterior (veremos abaixo) , em casos de ter várias tabelas envolvidas.

Junção exterior

Uma junção exterior [outer join] mostra todas as linhas de uma tabela, mesmo quando elas não satisfazem a condição de junção. Por exemplo:

```
select departamento.nome 'Departamento', funcionario.nome
'Funcionario'
from departamento left outer join funcionario
on departamento.coddepartamento = funcionario.coddepartamento
```

O resultado será algo semelhante a isso:

Se é usado LEFT OUTER JOIN indica que todas as linhas da tabela à esquerda (no caso, 'departamento') são incluídas no resultado. Nesse caso, são mostrados todos os departamentos, mesmo aqueles que não tem funcionário. Quando um departamento não tem funcionário, as colunas da tabela 'funcionario' irá mostrar o valor NULL . A tabela 'departamento' é chamada de a *tabela exterior* e 'funcionario' é a *tabela interior* da junção.
Se fosse usado RIGHT OUTER JOIN, a tabela à direita ('funcionario') mostraria todas as linhas e a tabela à esquerda, apenas as relacionadas. Se for usado FULL OUTER JOIN, todas as linhas de ambas as tabelas são incluídas, mesmo as que não estão relacionadas com a outra tabela.

Junções com mais de duas tabelas

Para falarmos de junções entre duas tabelas crie uma tabela chamada 'cargo', utilize o Enterprise Manager ou o comando Create table. Esta tabela terá as seguintes colunas:

ColumnName	**Datatype**	**Size**
CodCargo	int	
Nome	varchar	50

Ao criar esta tabela adicione os seguintes dados:

```
insert into cargo values (1, 'cargo1')
insert into cargo values (2, 'cargo2')
insert into cargo values (3, 'cargo3')
insert into cargo values (4, 'cargo4')
insert into cargo values (5, 'cargo5')
```

Na tabela 'funcionario' acrescente a seguinte coluna:

ColumnName	**Datatype**
CodCargo	int

Utilize o Enterprise Manager ou o comando Alter Table.
Vamos atualizar a tabela 'funcionario' para acrescentar dados a coluna cargo, execute os seguintes comandos:

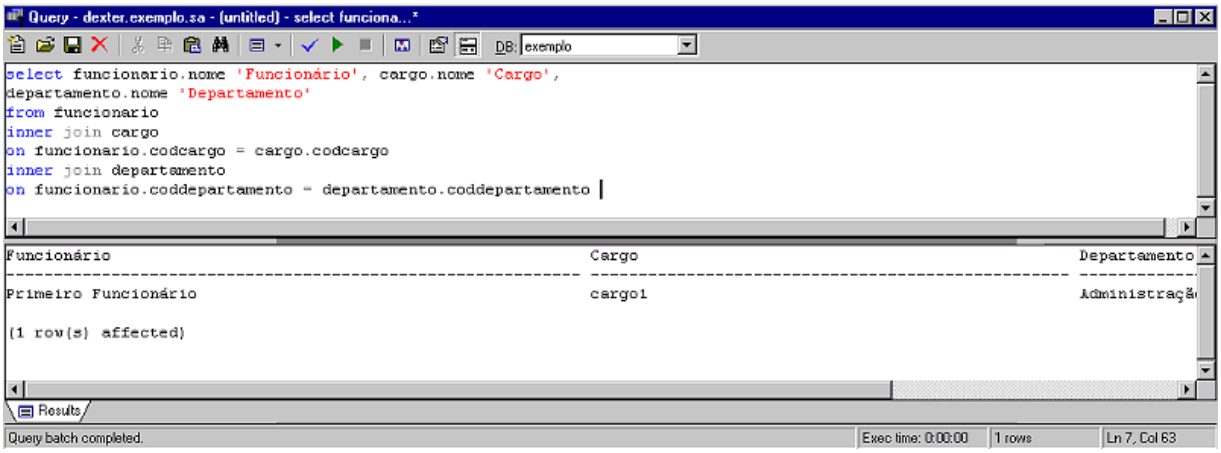
```
update funcionario set codcargo = 1 where codfuncionario = 1
update funcionario set codcargo = 2 where codfuncionario = 2
update funcionario set codcargo = 3 where codfuncionario = 3
update funcionario set codcargo = 4 where codfuncionario = 4
update funcionario set codcargo = 5 where codfuncionario = 5
```

É possível juntar três ou mais tabelas com informações relacionadas. No banco de dados 'Exemplos', por exemplo, a tabela 'cargo' contém uma linha que representa cada cargo do funcionário. Entre os dados, esta a identificação do cargo ('codcargo') . A coluna 'codcargo' pode ser usada para buscar o nome do cargo, fazendo uma junção com a tabela 'funcionario'. A tabela 'departamento' contém uma linha que representa cada departamento do funcionário. A coluna 'coddepartamento'pode ser usada para buscar o nome do departamento, fazendo uma junção com a tabela 'departamento'. Faremos o seguinte:

```
select funcionario.nome 'Funcionário', cargo.nome 'Cargo',
departamento.nome 'Departamento'
from funcionario
```

```
inner join cargo
on funcionario.codcargo = cargo.codcargo
inner join departamento
on funcionario.coddepartamento = departamento.coddepartamento
```

O resultado será semelhante a este:



Note que as duas junções de tabela são representadas com um INNER JOIN após o outro. O primeiro INNER JOIN cria uma "tabela" virtual reunindo 'funcionario' e 'cargo'. O segundo reúne essa tabela virtual à tabela 'departamento'.

Apelidos de tabela

Para simplicar a qualificação de colunas, pode-se usar um *apelido* [alias] de tabela, um nome colocado imediatamente após o nome da tabela, na lista do FROM. Esse nome representa a tabela nas qualificações. Por exemplo, a consulta anterior pode ser reescrita da forma:

```
select f.nome 'Funcionário', c.nome 'Cargo' , d.nome 'Departamento'
from funcionario f
inner join cargo c
on f.codcargo = c.codcargo
inner join departamento d
on f.coddepartamento = d.coddepartamento
```

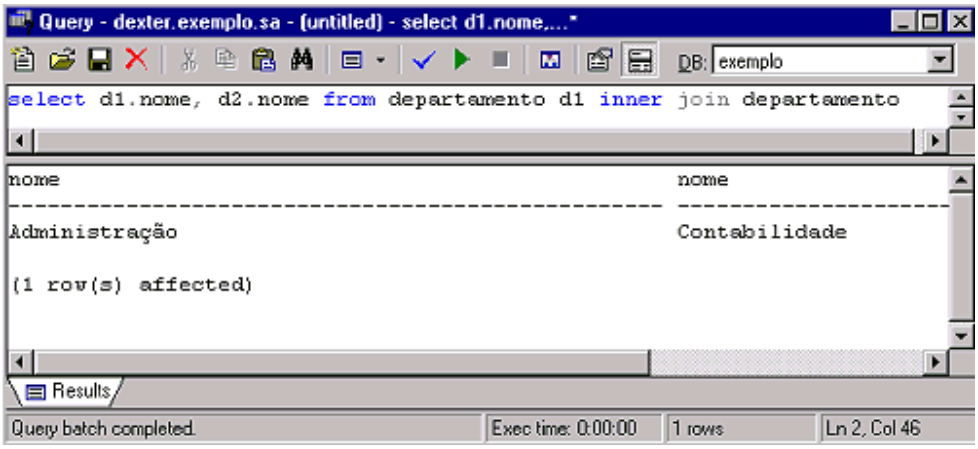
Nesse caso, 'f' é o apelido para a tabela 'funcionario', 'c' para a tabela 'cargo' e 'd' para a tabela 'departamento'. Note que os aliases podem ser usados na lista do SELECT ou nas condições de junção (ou em outros lugares, como numa cláusula WHERE).

Auto-junções

Uma *auto-junção* [self join] é uma junção da tabela com ela mesma. Na tabela departamento, por exemplo, cada departamento está subordinado a outro. A coluna 'coddeptsuperior' indica o código do departamento superior. Para mostrarmos uma lista de todos os departamentos, cada um com seus sub-departamentos, podemos usar:

```
select d1.nome, d2.nome from departamento d1 inner join departamento
d2 on d1.coddepartamento = d2.coddeptsuperior
```

Nesse caso, é obrigatório usar um apelido de tabela para distinguir as duas "cópias" da tabela que estão sendo relacionadas: 'd1' no exemplo representa uma linha da tabela 'departamento' e 'd2' representa outra linha que estão sendo comparadas entre si. O resultado dessa consulta é:



Subconsultas

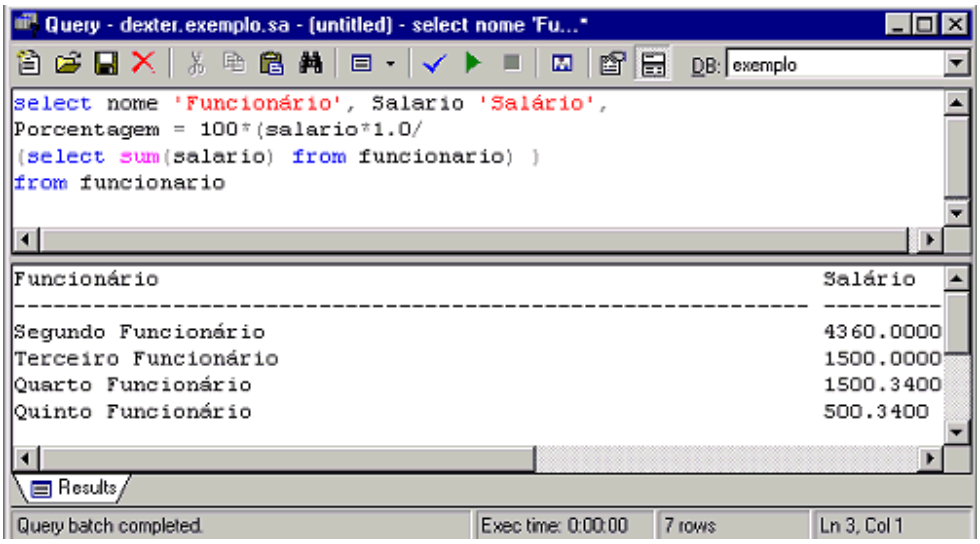
Uma *sub-consulta* [subquery] é uma consulta SELECT aninhada dentro de outro comando SQL. Ela pode retornar um valor só ou uma lista de valores para ser usada numa comparação. Por exemplo, para saber o gasto que a empresa tem com funcionários, pode-se usar:

```
select sum(salario) from funcionario
```

...que dá como resultado o valor 10.595,82 . Agora suponhamos que queremos saber qual a porcentagem do total que o salário do funcionário representa. Digite o seguinte comando:

```
select nome 'Funcionário', Salario 'Salário',
Porcentagem = 100*(salario*1.0/
(select sum(salario) from funcionario) )
from funcionario
```

O resultado será:



A coluna 'Porcentagem' é calculada como a quantidade dividida pelo resultado da sub-consulta (select sum(salario) from funcionario), que é = 10.595,82 nesse caso. Note portanto que a sub-consulta é executada e seu resultado é substituído dentro da outra consulta, como se tivéssemos escrito:

```
select nome 'Funcionário', Salario 'Salário',
Porcentagem = 100*(salario*1.0/ 10.595,82 )
from funcionario
```

Subconsultas são *sempre* colocadas entre parênteses e podem retornar no resultado apenas uma coluna (embora em alguns casos podem retornar mais de uma linha). No exemplo acima, se a subconsulta retornasse mais de um valor, haveria um erro de execução.

Subconsultas com operadores

Uma sub-consulta pode ser inserida nos resultados, como acima, ou pode ser usada numa expressão WHERE, com um operador de comparação, como =, <, >, <=, >= ou <>. Nesse caso ela deve retornar apenas um valor. Esse valor é substituído na consulta principal no momento da execução.

Listas de valores

Uma sub-consulta pode retornar uma lista de valores e essa lista de valores pode ser usada em comparações com o operador IN. Por exemplo, para saber quais são os funcionarios que possuem cargos, pode ser usado:

```
select funcionario.nome
from funcionario
where codcargo in (select codcargo from cargo)
```

Uma sub-consulta pode ser usada também com um operador de comparação modificado com as palavras ANY [qualquer] ou ALL [todos]. Por exemplo, > ALL [maior que todos] significa que para que a condição seja satisfeita, o valor comparado deve ser maior que todos os elementos da lista:
O exemplo abaixo utiliza o banco de dados Pubs.

```
select title
from titles
where advance > ALL
   (select advance
   from publishers, titles
   where titles.pub_id = publishers.pub_id
   and pub_name = 'Algodata Infosystems')
```

A subconsulta retorna uma lista de valores de 'advance' [adiantamento] que contém todos os valores de adiantamento de livros publicados pela editora 'Algodata Infosystems'. Seu resultado é (5000, 5000, 5000, 7000, 8000) A consulta externa retorna os livros cujo 'advance' é maior do que todos os itens dessa lista.
As seguintes combinações que podem ser usadas:

> ALL	maior que todos os elementos da lista
< ALL	menor que todos
<> ALL	diferente de todos (o mesmo que NOT IN)
= ANY	igual a *algum* dos elementos da lista (o mesmo que IN)
> ANY	maior que algum dos elementos da lista
< ANY	menor que algum dos elementos
<> ANY	diferente de algum dos elementos da lista (falso se igual a todos)

Além disso, podem ser usadas combinações com >= e <=, de forma análoga. Note que não é permitido o uso de = ALL.

Testes de existência

Um teste de existência é uma condição que envolve a palavra EXISTS e uma sub-consulta. A condição é verdadeira se a sub-consulta retorna alguma linha e é falsa se ela retorna zero linhas. Por exemplo, para saber quais os departamentos que possue funcionários cujo cargo é igual a cargo1, utilize o banco de dados Exemplos e execute o seguinte comando:

```
select d.nome
from departamento d
where exists (select *
    from funcionario f, cargo c
    where f.codcargo = c.codcargo
       and c.nome = 'Cargo1'
       and d.coddepartamento = f.coddepartamento)
```

O resultado da sub-consulta não importa, pois está apenas sendo testada a existência de um resultado. Nesse caso, a lista de colunas é sempre um asterisco (*).

Subconsultas correlacionadas

As sub-consultas que foram vistas até agora nos exemplos podem ser avaliadas uma vez só e depois substituídas no corpo da consulta principal. Já uma sub-consulta *correlacionada* [correlated subquery] depende dos valores da consulta principal onde ela está alinhada, por isso deve ser avaliada uma vez para cada linha do resultado externo.
Por exemplo, a seguinte utilize o banco de dados Pubs para consultar a lista, para cada livro, o autor que tem a maior porcentagem de royalties sobre o livro (a tabela 'titleauthor' relaciona livros e autores de forma N x N):

```
select title_id, au_id, royaltyper
from titleauthor ta
where royaltyper = (select max(royaltyper)
     from titleauthor
     where title_id = ta.title_id)
```

Essa é uma sub-consulta correlacionada porque ela faz referência a uma tabela da consulta mais externa. A sub-consulta é avaliada repetidas vezes, uma para cada linha da tabela 'titleauthor'.

9 - Implementando Índices

Por que índices?

Tipos de Índices

Otimizando Consultas

Objetivos:
- Aprender a criar índices;
- Entender o funcionamento do otimizador de consultas.

Por que índices?

Índice [index] é um mecanismo que acelera bastante o acesso aos dados. Consiste de uma estrutura de dados que contém ponteiros ordenados para os dados. O SQL Server usa índices

[indexes ou indices] automaticamente em várias situações para acelerar a pesquisa e atualização de dados, como por exemplo onde houverem restrições [constraints] PRIMARY KEY e UNIQUE.
Recomenda-se considerar o seguinte para a criação de índices:

- Se uma coluna está presente na cláusula WHERE em um comando SELECT, UPDATE ou DELETE, o SQL Server consegue verificar as condições mais rapidamente se houver um índice. Caso contrário, ele faz uma varredura seqüencial da tabela [table scan].
- Se uma coluna é muito usada para ordenar valores (com ORDER BY), essa ordenação é muito mais eficiente se ela tiver um índice.
- Se uma coluna é usada para fazer junção de duas tabelas, essas junções podem ser feitas mais eficientemente se ela estiver *indexada*.

Índices não apenas aceleram a recuperação de linhas em consultas, mas eles também aumentam a velocidade de atualizações e exclusões. Isso ocorre porque o SQL Server deve encontrar uma linha, antes de poder atualizá-la ou excluí-la. No entanto, índices levam tempo para serem criados e ocupam espaço em disco. Cada atualização na tabela também atualiza dinamicamente todos os índices definidos. Portanto, se você criar muitos índices inúteis numa tabela, pode estar atrapalhando o desempenho da atualização de dados sem agilizar muito o tempo de resposta nas consultas.
No geral, o aumento da eficiência obtido com o uso de índices para localizar a linha sobrepuja a carga extra de trabalho necessária para atualizar os índices, a não ser, como mencionado acima, que a tabela tenha muitos índices.

O Otimizador

O otimizador é o componente do SQL Server que analisa as consultas SQL e decide quando vale a pena utilizar um índice ou não. Às vezes, mesmo quando você define um índice em uma coluna, o otimizador resolve não utilizá-lo por determinar que ele não ajudaria no desempenho. Por exemplo, não vale a pena utilizar um índice que retorna uma porcentagem muito grande de linhas, pois levaria mais tempo analisando o índice do que o tempo que ele economizaria filtrando os resultados. Por exemplo, se uma coluna tem apenas três valores possíveis, 0, 1, e 2, não vale a pena indexar, pois qualquer consulta pode retornar até 33% das linhas. O otimizador descobre isso e ignora esse tipo de índice.

Tipos de Índices

Clustered

Um índice *clustered* [agrupado] é aquele onde a ordem física das páginas de dados é a mesma ordem do índice. A cada inserção, numa tabela que tem um índice agrupado, a ordem física dos dados pode mudar. Só pode haver um único índice agrupado por tabela.Se você não especificar o indice Clustered a sua tabela será criada com o índice Non-clustered (ver abiaxo).
Recomenda-se criar um índice agrupado antes de qualquer outro, pois ao criá-lo, as linhas da tabela são reordenadas fisicamente e todos os outros índices são reconstruídos.
É recomendável usar um índice agrupado para a coluna que representa a *ordem mais natural* da tabela, ou seja, a ordem na qual geralmente os resultados serão apresentados.
Recomenda-se utilizar índice agrupado [Clustered] nos seguintes casos:

- Os dados das colunas são acessados frequentemente. Por exemplo na tabela de 'funcionario'do banco de dados Exemplo , vamos supor que são feitas várias pesquisas com o nome do funcionario, neste caso você poderia criar um índice clustered com o nome do funcionário.
- Em colunas usadas com ORDER BY e GROUP BY.
- Em colunas que são alteradas frequentemente.
- Em chaves primárias, contanto que não haja outras colunas melhores.
- Em chaves estrangeiras, porque geralmente elas não são únicas.

Non-clustered

Um índice *non-clustered* [não-agrupado] possui uma ordem física diferente da ordem dos dados. Existe um nível a mais, de *ponteiros* para os dados, que permite acessá-los indiretamente.
Pode haver mais de um índice não-agrupado na tabela, até o máximo de 249 índices, incluindo qualquer índice criado com restrições PRIMARY KEY ou UNIQUE.
Quando o tipo de índice da tabela não for específicado ele será criado como um índice Nonclustered.
É recomendado utilizar índices não agrupados [Nonclustered] para:

- Colunas que são usadas nas claúsulas ORDER BY e GROUP BY.
- Colunas que são frequentemente utilizadas como condições na claúsula WHERE.

Características dos Índices

Único

Um índice *único* [unique] é aquele onde os valores da chave não podem ser repetidos, ou seja, os valores das colunas do índice, tomados em conjunto, não podem se repetir. Um índice único pode ser agrupado ou não-agrupado.
Por exemplo, na tabela Cliente, poderia ser criado um índice único para a coluna CodCliente, significando que não pode haver valores duplicados nessa coluna. Se você tenta inserir dados em uma tabela com valores repetidos para CodCliente, a inserção falha.
Quando da criação de um índice único, não pode haver valores duplicados nas colunas do índice. Se houver, a criação do índice falha e você deve alterar as colunas antes de tentar criá-lo novamente.

Composto

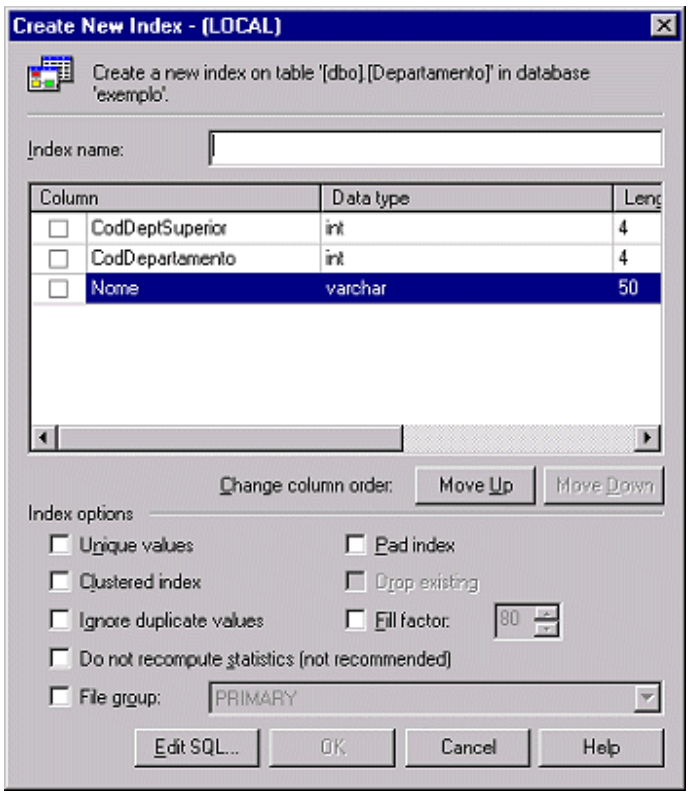
Um índice composto é aquele formado por duas ou mais colunas. Esse tipo de índice é útil quando duas ou mais colunas são sempre pesquisadas em conjunto. Por exemplo, poderia ser criado um índice na tabela Cliente para as colunas (Cidade,Estado). A ordem das colunas importa: um índice com (Estado,Cidade) seria completamente diferente.
Se você criar um índice composto, o otimizador vai utilizá-lo mesmo quando apenas a primeira coluna é especificada, por exemplo em:

```
SELECT * FROM Cliente WHERE Cidade = 'Goiânia'
```

Um índice composto também pode ser único. Nesse caso, o que não pode se repetir é o valor das duas ou mais colunas, tomadas em conjunto. Por exemplo, os valores poderiam ser (1,1), (1,2), (2,1), (2,2) etc. Mas não poderia haver duas linhas com os valores (1,1).

Criando e excluindo índices utilizando o Enterprise Manager

Um índice pode ser criado no Enterprise Manager. Basta clicar na tabela desejada com o botão direito, selecionar **All Tasks** e **Manage Indexes**.

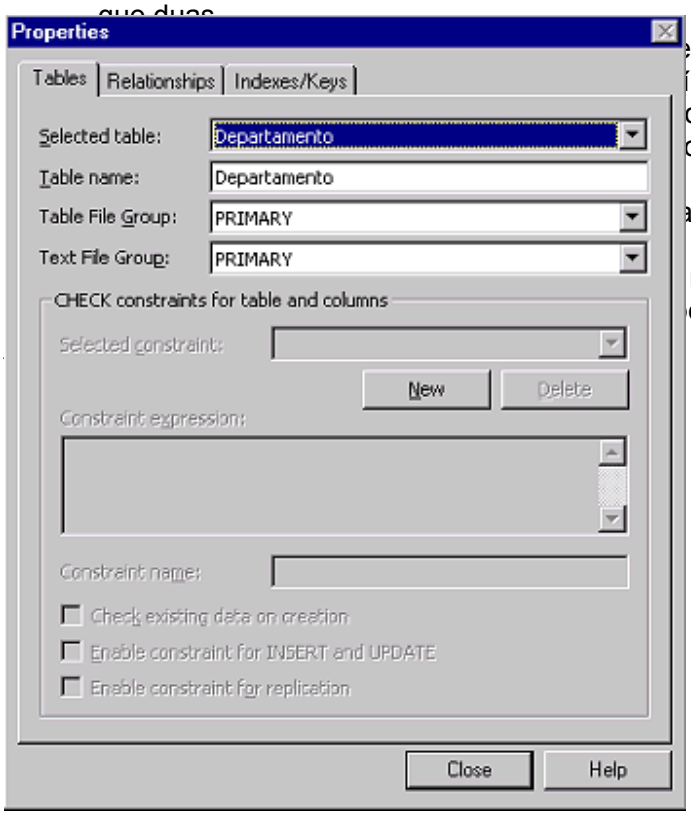


Na opção 'Table' informe a tabela em que deseja criar o índice. Clique no botão **New...**. Aparece a tela abaixo:

Em 'Column', marque na caixa de verificação a(s) coluna(s) que você quer que faça(m) parte do índice. Você pode mover qualquer coluna selecionada para cima ou para baixo (lembrando que em um índice ocmposto a colunas em ordens diferentes formam índices diferentes)
Em 'Index name:' coloque o nome do índice que deseja criar ou visualizar.
As próximas opções irão definir o tipo e as características do índice:
Em 'Index options', pode-se definir o seguinte, com a seleção das opções adequadas:

- se você marcar "Unique values" [valores únicos], o índice será um índice único.
- No caso de um índice único, se você marcar a opção "Ignore Duplicate values" [ignorar valores duplicados], ao executar um comando INSERT ou UPDATE em várias linhas, apenas as linhas que não têm chave duplicada serão inseridas/atualizadas; para as outras, o SQL Server mostra uma mensagem de aviso e ignora (não insere) as linhas duplicadas. Se esta opção não for marcada, o comando INSERT ou UPDATE falha e não insere/atualiza nenhuma linha.
  Nota: Um índice único não pode ser criado em uma coluna que já tenha valores duplicados, mesmo que você selecione a opção de "ignorar valores duplicados". Se você tentar, o SQL Server mostra uma mensagem de erro e lista os valores duplicados. Elimine os valores duplicados antes de criar um índice único na coluna.

- Para criar um índice agrupado, marque a opção "Clustered". Com o índice agrupado, você não pode ignorar valores duplicados (é permitido marcar a opção, mas ao clicar em OK, surge uma mensagem de erro, avisando que as duas opções são mutuamente exclusivas). Porém, se você marcar, junto com a opção "Clustered", a opção "Unique values", é permitido ignorar valores duplicados (selecionando "Ignore Duplicate Values", que serão tratados da mesma forma que no caso do índice ser apenas único (com a diferença que agora ele é um índice único agrupado).
  Se já houver um índice agrupado na tabela, na criação de um novo índice, não se permite selecionar a opção "clustered"(pois só pode haver um índice agrupado em qualquer tabela).
- "Do not recompute statistics": com esta opção marcada, as estatísiticas do índice não são recalculadas automaticamente quando o índice é atualizado.
- Filegroup: especifica em que grupo de arquivos será criado o índice. Clique no nome do grupo de arquivos. (veja grupos de arquivos)
- "Drop existing": para excluir qualquer índice existente com o mesmo nome antes de criar o novo índice. Utilizada quando você altera um índice já criado. Você não pode marcar ou desmarcar essa opção.
  Nota: Você não pode transformar um índice agrupado em um índice não agrupado, editando-o. Você deve excluí-lo, para então criar outro índice (que pode ter o mesmo nome).
- "Fill factor": (fator de preenchimento) especifica o quão cheio o SQL Server deve fazer o nível folha de cada página de índice durante a criação do mesmo.Quando uma página de índice fica cheia, o SQL Server deve gastar tempo para dividir a página, liberando espaço para novas colunas, o que tem um custo computacional alto. Para tabelas em que são feitas muitas atualizações, um valor para "Fill factor" bem escolhido gera um melhor desempenho em atualizações do que um valor mal escolhido. O valor de "Fill factor" é fornecido na forma de porcentagem.
- "Pad index": Especifica o espaço a ser deixado desocupado em cada página nos níveis intermediários do índice. Só é útil quando selecionado em conjunto com "Fill factor", pois o "pad index" usa a porcentagem definida em "Fill factor". Independentemente do valor de "Fill factor", o número de colunas numa página intermediária nunca é menor do



e o índice será criado (ou alterado). Você índice, bastgando para isso clicar no de verificá-lo antes de executar, bastando o mesmo efeito de clicar em Ok na tela

ager, basta selecioná-lo, na tela "Manage

no modo de edição da tabela, selecionar erties, na janela que aparece, que é a

Criando e excluindo índices com comandos SQL

Também é possível criar um índice com o comando SQL CREATE INDEX e excluir um índice com DROP INDEX.

```
CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED]
    INDEX nome_indice ON tabela (nome_coluna1 [, nome_coluna2, ...n])
[WITH
        [PAD_INDEX]
        [[,] FILLFACTOR = fator_preenchimento]
        [[,] IGNORE_DUP_KEY]
        [[,] DROP_EXISTING]
        [[,] STATISTICS_NORECOMPUTE]
]
[ON grupo_arquivos]
```

Onde:
nome_indice é o nome do índice que deseja criar.
nome_tabela é o nome da tabela que deseja criar o índice.
nome_coluna1 é o nome da coluna que irá fazer parte do índice.Se o índice tiver mais de uma coluna acrescente a vírgula e coloque o nome das outras colunas.
UNIQUE indica se o índice será único. É opcional. Se o índice fo único você pode acrescentar a opção IGNORE_DUP_KEY [ignorar chaves duplicadas].
CLUSTERED indica se o índice será agrupado. Com o índice agrupado e a opção UNIQUE, você pode também usar a opção IGNORE_DUP_KEY,
FILLFACTOR é o fator de prenchimento, ou seja, a porcentagem de espaço livre que será deixado em cada página do índice.
DROP_EXISTING exclui o índice existente com o mesmo nome. Se você for criar um índice cujo nome não existe e usar esta oção, o SQL Server retornará uma mensagem avisando que o índice com o nome sendo criado não foi encontrado.
STATISTICS_NORECOMPUTE faz com que as estatísticas do índice não sejam recalculadas automaticamente com a atualização do índice.
PAD_INDEX deixa espaços vazios nas páginas dos níveis intermediários do índice. Só faz sentido se usado em conjunto com FILLFACTOR.
Nota: Para criar qualquer índice, você deve estar posicionado no banco de dados em que o mesmo será criado, ou informar o nome completo da tabela (nome_banco_de_dados..nome_tabela). Também é possível usar a cláusula USES antes do comando de criação do índice (USES nome_banco_de_dados).
Para excluir algum índice, use o comando DROP INDEX, com a seguinte sintaxe:

```
DROP INDEX 'tabela.índice' [,...n]
```

Onde tabela.indice é o nome da tabela, seguido do nome do índice que se deseja excluir. Caso você queira excluir mais de um índice de uma vez, basta colocar uma vírgula e indicar o nome do(s) outro(s) índice(s) a ser(em) excluído(s).

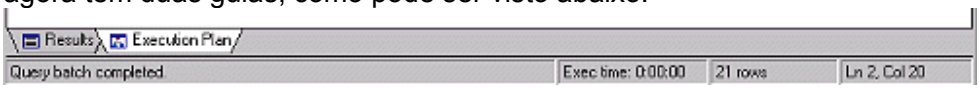
Otimizando Consultas

O otimizador escolhe uma de duas alternativas ao fazer uma consulta: ou varre a tabela ou usa um índice. Ele decide o que fazer baseado em:

- Estimativa aproximada de quanto trabalho é necessário para usar o índice ou não. Essa estimativa é baseada em informações estatísticas sobre o índice, que dizem qual a distribuição dos dados. Essas informações podem ficar desatualizadas. Para corrigi-las, execute o comando UPDATE STATISTICS *nome_da_tabela*.
- Se uma tabela é muito pequena, ou se o otimizador espera que será retornada uma grande porcentagem de linhas, ele faz uma varredura.
- Se na cláusula WHERE da consulta estão presentes colunas indexadas, é bem provável que o otimizador resolva utilizar o índice.

Analisando o Otimizador

Para saber se o otimizador está usando seus índices ou não, no Query Analyzer, clique no botão "Show Execution Plan" (Ctr+L). Quando da execução de uma consulta, você agora tem duas guias, como pode ser visto abaixo:



Essa guias podem ser selecionadas na parte inferior da janela da consulta. Uma é o "Estimated Execution Plan" e a outra "Results". Na guia "Estimated execution plan", você pode ver a análise de cada linha da sua consulta ou qualquer comando SQL. Para cada linha da consulta, você verá seu custo "Query cost" como uma porcentagem do custo total da sequência de comandos, e seu texto "Query text". Na guia "Results" você verá o resultado da consulta.
Além disso, para cada linha da consulta, é mostrada uma representação gráfica (que deve ser lida da direita para a esquerda), que especifica os operadores lógicos e físicos utilizados na execução de cada parte da consulta ou comando. Para saber mais sobre a representação gráfica do plano de execução, procure por"Graphically Displaying the Execution Plan Using SQL Server Query Analyzer" no Books Online.
Cada um dos ícones (que são chamados de operadores, físicos e lógicos) mostrados no plano de execução, quando se passa o mouse por cima dos mesmos, mostram informações específicas a seu respeito, como seu nome, o custo computacional em termos de CPU e de I/O, além dos parâmetros que foram usados com o mesmo e uma breve descrição de sua função.
Quando você passa o mouse por cima da seta que liga os ícones, você vê quantas linhas foram retornadas (no caso de um SELECT) pelo seu comando, e o tamanho estimado de cada linha.
Caso você queira ver os resultados da análise do otimizador em modo texto, use o comando
`SET SHOWPLAN_ALL ON`
Isso faz com que o SQL Server não execute comandos SQL. Ao invés disso, ele retorna informações detalhadas sobre como os comandos são executados e estima os custos dos comandos. A informação é retornada como um conjunto de linhas que formam uma árvore hierárquica que representa os passos dados pelo processadore de consultas do SQL Server na

execução de cada comando, seguida por algumas linhas com os detalhes dos passos de execução.
Esse comando não pode ser parte de um procedimento armazenado; deve ser o único comando em um batch (lote de comandos). O comando é destinado a aplicações escritas para lidar com sua saída. Para retornar saídas compreensíveis para aplicações MS-DOS, use
```
SET SHOWPLAN_TEXT ON
```
Os resultados são retornados na forma de análise da consulta, sem sua execução, até que você "desligue" essas opções, bastando para isso digitar
```
SET SHOWPLAN_ALL OFF
```
ou
```
SET SHOWPLAN_TEXT OFF
```
dependendo de qual das opções estiver "ligada"

10 - Integridade de Dados

Conceitos

A propriedade IDENTITY

Usando Defaults e Regras

Definindo e Usando Restrições [Constraints]

Quando Usar Cada Componente

Objetivos:
- Aprender a criar colunas com auto-incremento;
- Aprender a utilizar as opções defaults e Regras nas colunas;
- Aprender a definir restrições nas tabelas para garantir a integridade dos dados;
- Saber quando escolher cada um dos recursos de integridade de dados.

A propriedade IDENTITY

Uma coluna criada com a propriedade IDENTITY tem um valor único que é gerado automaticamente pelo sistema. Somente uma coluna pode ter essa propriedade. Por exemplo, crie uma nova tabela no banco de dados Exemplo, com o seguinte comando:
```
create table Produto
(CodProduto int NOT NULL IDENTITY,
  Nome varchar(60),
  Preço money)
```
Uma coluna IDENTITY não aceita um valor explicitamente inserido. Ao inserir dados na tabela, a coluna deve ser omitida. Execute agora:
```
insert into Produto (Nome, Preço)
  values ('Primeiro Produto', 100.0)
insert into Produto (Nome, Preço)
  values ('Segundo Produto', 150.0)
insert into Produto (Nome, Preço)
  values ('Terceiro Produto', 120.0)
```

Execute agora:

```
select * from Produto
```

Note que a coluna CodProduto foi preenchida automaticamente com um valor auto-incrementado pelo sistema:

```
CodProduto        Nome                      Preço
-----------       ------------------        --------------------------
1                 Primeiro Produto          100,00
2                 Segundo Produto           150,00
3                 Terceiro Produto          120,00
(3 row(s) affected)
```

Opcionalmente, na criação da tabela, pode ser informado uma *semente* (valor inicial para a coluna) e um *incremento*, como em:

```
CodProduto int IDENTITY(0,10)
```

Onde o 0 é a *semente* e o 10 é o incremento.

Desabilitando IDENTITY

Você pode temporariamente desativar a propriedade IDENTITY, para que você possa inserir valores explicitamente numa coluna com IDENTITY. Pode ser necessário que você insira valores explicitamente em tabelas que têm itens deletados com frequência.Inserir valores explicitamente na coluna com IDENTITY lhe permite preencher espaços vazios deixados na tabela.

Para desativar a geração automática de valores, use:

```
set identity_insert nome_da_tabela on
```

Para voltar ao funcionamento normal, use:

```
set identity_insert nome_da_tabela off
```

A qualquer instante, somente *uma tabela* em uma sessão pode ter a propriedadede de IDENTITY_INSERT em ON. Esta propriedade só é válida para o usuário atual e a sessão atual (perde o efeito quando você se desconecta do SQL Server).

Se já houver uma tabela com esta propriedade em ON, e se entrar com o comando SET IDENTITY_INSERT ON para outra tabela, será retornado uma mensagem de erro dizendo que essa propriedade já está em ON e qual a tabela para a qual essa propriedade está em ON.

Identificador globalmente exclusivo (GUID)

Embora a propriedade IDENTITY automatize a numeração de colunas dentro de uma tabela, tabelas separadas, cada uma com sua própria coluna de identificador, pode gerar os mesmos valores. Isso ocorre porque se garante que a propriedade IDENTITY seja única apenas para a tabela na qual ela for usada. Se uma aplicação deve gerar uma coluna de identificador que seja única em todo o banco de dados ou em todos bancos de dados de cada computador ligado em rede no mundo, use a propriedade ROWGUIDCOL, o tipo de dados **uniqueidentifier**, e a função NEWID.

O tipo de dados **uniqueidentifier** armazena valores binários de 16 bits que operam como números globalmente exclusivos de identificação (GUID). Um GUID é um número binário que se garante ser exclusivo; nenhum outro computador no mindo gerará uma cópia daquele valor GUID. A principal utilidade de um GUID é a atribuição de um identificador que deva ser exclusivo em uma rede que tenha diversos computadores em diversos locais.

Quando você usar a propriedade ROWGUIDCOL para definir uma coluna com identificador globalmente exclusivo, considere que:

- Uma tabela só pode ter uma coluna ROWGUIDCOL, e essa coluna deve ser definida utilizando o tipo de dados uniqueidentifier.

- O SQL Server não gera automaticamente valores para a coluna. Para inserir um valor globalmente exclusivo, crie uma definição DEFAULT na coluna que usa a função NEWID para gerar um valor globalmente exclusivo.
- Como a propriedade ROWGUIDCOL não força a unicidade, a restrição UNIQUE deve ser usada para se assegurar que valores exclusivos sejam inseridos na coluna ROWGUIDCOL.

O tipo de dados **uniqueidentifier** não gera automaticamente novos IDs para colunas inseridas, como a propriedade IDENTITY o faz. Para obter novos valores **uniqueidentifier**, uma tabela deve ter uma cláusula DEFAULT especificando a função NEWID, ou os comandos INSERT devem usar a função NEWID. Por exemplo:

```
CREATE TABLE TabelaExclusiva
  (ColunaExclusiva UNIQUEIDENTIFIER   DEFAULT NEWID(),
   Caracteres VARCHAR(10))
GO
INSERT INTO TabelaExclusiva(Caracteres) VALUES ('abc')
INSERT INTO TabelaExclusiva VALUES (NEWID(), 'def')
GO
```

A principal vantagem do tipo de dados **uniqueidentifier** é que se garante que os valores gerados pela função Transact-SQL, NEWID, sejam exclusivos ao redor do mundo.
Mas por outro lado, o tipo de dados **uniqueidentifier** tem sérias desvantagens:

- Os valores são longos e obscuros. Isso os torna difíceis de serem digitados corretamente pelos usuários, e mais difícil ainda de serem lembrados.
- Os valores são aleatórios e não aceitam quaisquer padrão que os torne mais significativos para os usuários.
- Não há como determinar a sequência em que os valores **uniqueidentifier** são gerados. Eles não se adequam a aplicações existentes que incrementem serialmente valores-chave.
- Como têm 16 bytes, os dados do tipo **uniqueidentifier** são relativamente grandes se comparados com outros tipos de dados tais como inteiros de 4 bytes. Isto significa que índices construídos usando chaves do tipo **uniqueidentifier** podem ser relativamente mais lentos do que se implementados utilizando uma chave **int**.

Usando Defaults e Regras

Um *default* é um valor que é usado para colunas quando seus valores não são explicitamente informados. Um default pode ser criado como um objeto à parte ou como *restrição* de uma coluna, como veremos mais tarde.
Uma *regra* é uma condição que é verificada quando dados são inseridos numa tabela. Ela também pode ser criado como um objeto à parte ou como uma restrição CHECK, como veremos.

Criando e utilizando um Default

Para criar um default no Enterprise Manager, clique no item "Defaults" com o botão direito e em **New Default**. Digite o nome do default, nesse caso 'PrecoDefault'. Em 'Description', digite o valor dele, que será 50.00 e clique no botão Ok.
Para vincular um default a uma coluna, de forma que ela passe a usar esse valor default, selecione o default que você acabou de criar, clique nele com o botão direito e em Properties. Clique no botão "Bind Columns". Selecione a tabela, no caso "Produto". Clique em Preço, na coluna "Unbound Columns" e clique em Add>>. A coluna que você tiver selecionado aparece agora do lado direito, em baixo de Bound Columns. Clique em Apply. Feche todas as janelas clicando em Ok.
Para testar o default, insira valores na tabela "Produto" sem informar o preço:

```
insert into Produto (Nome) values ('Produto default')
```

Verifique o conteúdo da tabela. O 'Produto default' deve ter o Preço=50.00.

Criando e Utilizando uma Regra

Uma regra verifica o valor de uma coluna para saber se esse valor será aceito ou não. Se um valor inserido com INSERT ou atualizado com UPDATE não satisfaz a regra, ocorre um erro e a operação é cancelada. Uma regra contém uma condição qualquer (semelhante a uma cláusula WHERE) que tem um *parâmetro* a ser verificado. Esse parâmetro é substituído pelo valor da coluna no momento de execução da regra.
Um parâmetro é sempre iniciado com @ e pode ter qualquer nome. Ele pode ser usado mais de uma vez no texto da regra, se necessário.
Vamos criar uma regra para verificar se um estado é válido. Ela irá verificar se o valor informado pertence a um conjunto de siglas válidas de estado. No Enterprise Manager, clique em "Rules" com o botão direito e em **New Rule**. Digite no nome da regra "RegraEstado" e em "text", digite:

```
@valor in ('GO', 'TO', 'RJ', 'SP')
```

Note que para simplificar não colocamos todos os estados válidos. O nome do parâmetro é valor, e a condição verifica se @valor é um dos valores da lista. Clique no botão Ok.
Agora essa regra pode ser usada em uma coluna qualquer. Clique com o botão direito na regra que você acabou de criar, selecione Properties, clique no botão "Bind Columns"e vamos ligar essa regra, na tabela Cliente, à coluna "Estado". Faça isso, selecionando a tabela Cliente, selecionando o campo Estado em "Unbound Columns", e clicando no botão Add. Feche as janelas, clicando em Ok duas vezes. Note que a regra só se aplica aos novos dados que serão inseridos e não afeta os anteriores.
Agora tente inserir um dado na tabela Cliente, como por exemplo:

```
insert into Cliente (CodCliente, Nome, Estado)
values (10, 'Décimo Cliente', 'XY')
```

Como 'XY' não satisfaz a regra, o SQL Server vai mostrar uma mensagem indicando isso.

Vinculando a tipos de dados

Um default ou uma regra pode ser vinculado(a) a um tipo de dados definido pelo usuário. Nesse caso, todas as colunas que forem criadas com aquele tipo terão o mesmo valor default (caso não tenha sido especificado um outro, que nesse caso tem precedência) ou a mesma regra de validação.
Para testar isso, vamos criar um novo tipo de dados chamado 'estado'. Clique com o botão direito em "User defined data types" e em **New User Defined Data Type**. Chame o tipo de 'estado' e na sua descrição coloque CHAR(2). Na coluna "Default", você pode selecionar um valor default, e o SQL Server deixa você selecionar mesmo valores incompatíveis.

Se uma coluna tem um default e uma regra associados a ela, o valor default não pode violar a regra. Um default que conflite com uma regra nunca é inserido, e a cada vez que se tentar inserir o default, o SQL Server gera uma mensagem de erro.
Se você selecionar um default que use um dado de tipo incompátivel com o tipo de dados da coluna, quando tentar inserir dados nessa coluna, será inserido o valor NULL na mesma. Mas, se a coluna não aceitar valroes NULL, o SQL Server reportará um erro na hora de tentar inserir valores nessa coluna, que tentem usar o default
Logo, não selecione nenhum valor para Default. Na coluna "Rule", selecione "RegraEstado".
Se for criada uma nova tabela, com uma coluna que utiliza o tipo 'estado', ela terá essa verificação da "RegraEstado", automaticamente.

Usando comandos SQL

Um default também pode ser criado com o comando CREATE DEFAULT, como:
```
create default PrecoDefault as 50.00
```
Uma regra pode ser criada com o comando CREATE RULE, como:
```
create rule RegraEstado as @estado in ('SP', 'GO', 'RJ')
```
Para vincular uma regra ou default a uma coluna ou a um tipo de dados, usa-se o procedimento *sp_bindrule* ou *sp_bindefault*:
```
sp_bindefault nome_default, nome_objeto, futureonly
sp_bindrule nome_regra, nome_objeto, futureonly
```
Onde *nome_objeto* pode ser *nome_tabela.nome_coluna*, no caso de uma coluna ou o nome de um tipo de dados e **futureonly** é um parâmetro opcional dizendo que, no caso de um tipo de dados, o item afeta apenas colunas a serem criadas, mas não as existentes.
Para desvincular a regra ou default, usa-se:
```
sp_unbindefault nome_objeto, futureonly
sp_unbindrule nome_objeto, futureonly
```
Ao desvincular um default ou uma regra, se você usar a opção **futureonly**, colunas existentes do tipo de dados não perdem o default ou regra especificado.
Finalmente, para excluir um default ou regra, podem ser usados os comandos:
```
drop default nome_default
drop rule nome_regra
```
Lembre-se de desvincular um default ou regra que estejam vinculados a uma coluna antes de excluí-los.

Definindo e usando restrições [constraints]

Uma *restrição* [constraint] é uma propriedade de uma coluna usada para reforçar a integridade de dados. Geralmente restrições são definidas quando a tabela é criada (com CREATE TABLE), mas podem também ser definidas ou retiradas quando a tabela já contém dados (com o comando ALTER TABLE). Se um comando de alteração (INSERT ou UPDATE) não satisfaz uma das restrições, o comando é cancelado.
Toda restrição tem um nome, que você pode informar nos comandos CREATE TABLE e ALTER TABLE. Se você não informar um nome, o SQL gera um automaticamente, como PK_titleauth_au_id_154Af3e0.
De forma geral, a sintaxe para uma restrição é:
CONSTRAINT *nome_da_restrição definição*
Onde a *definição* inicia com as palavras PRIMARY KEY, UNIQUE, CHECK, FOREIGN KEY ou DEFAULT. A palavra CONSTRAINT e o *nome_da_restrição* podem ser omitidos. Nesse caso, o nome será gerado automaticamente.

Chave primária [PRIMARY KEY]

A *chave primária* [primary key] de uma tabela é uma coluna ou seqüência de colunas que identificam unicamente uma linha dentro da tabela, ou seja, seu valor não pode ser repetido para outras linhas. Ao definir uma chave primária, automaticamente é criado um índice na tabela. Só pode haver uma chave primária na tabela. Não se pode entrar com um valor nulo em qualquer coluna de uma chave primária (lembrando que nulo é um valor desconhecido, diferente de 0 ou de espaço em branco). Recomenda-se uma coluna inteira, pequena, como uma chave primária.
Na criação da tabela, essa restrição pode ser definida da seguinte forma (quando a chave primária é composta de uma só coluna):

```
create table Fornecedor (
  CodFornecedor int not null primary key,
  Nome varchar(50) null,
  Endereco varchar(50) null,
  Telefone varchar(20) null
)
```

Note que 'CodFornecedor' *deve* ter a opção NOT NULL. Não é possível criar uma chave primária com colunas que podem ser NULL. Opcionalmente, poderia ser informado um nome para a restrição, por exemplo 'ChaveFornecedor'. Nesse caso, a segunda linha acima seria:

```
CodFornecedor int not null constraint ChaveFornecedor primary key,
```

Agora, na tabela Fornecedor, não pode haver duas linhas com o mesmo valor de 'CodFornecedor'.
Quando a chave é composta de duas ou mais colunas, nesse caso ela tem que ser especificada com a lista de colunas entre parênteses, por exemplo:

```
create table ProdutoFornecedor
(CodProduto int, CodFornecedor int,
primary key (CodProduto, CodFornecedor))
```

Uma chave primária pode ser acrescentada à tabela depois que ela já foi criada, com o comando ALTER TABLE. Por exemplo, vamos acrescentar chaves primárias às tabelas Cliente e Produto:

```
alter table Cliente
  add primary key nonclustered (CodCliente)
alter table Produto
  add primary key clustered (CodProduto)
```

A opção NONCLUSTERED diz respeito ao tipo de índice que será criado para a chave primária. Se não especificada, o índice será CLUSTERED (v. capítulo anterior). Esse índice é criado ou excluído automaticamente, junto com a restrição.
Em qualquer um dos casos pode-se especificar ou não o nome da restrição, na forma CONSTRAINT *nome* logo antes das palavras "PRIMARY KEY".

Unicidade [UNIQUE]

Uma restrição UNIQUE em uma coluna ou grupo de colunas determina que o seu valor deve ser único na tabela. Esse tipo de restrição é usado para *chaves alternadas*, ou seja, valores que se repetem na tabela além da chave primária. Pode haver várias restrições UNIQUE na tabela e as colunas de uma restrição UNIQUE permitem valores nulos.
Esse tipo de restrição pode ser criada com exatamente a mesma sintaxe do PRIMARY KEY, por exemplo (não execute):

```
alter table Cliente
  add unique nonclustered (CodCliente)
```

Também é criado um índice automaticamente, que não permite valores duplicados. Pode-se especificar ou não o nome da restrição, na forma CONSTRAINT *nome* logo antes do "UNIQUE"..

Default

Um default pode ser especificado na forma de restrição. Na definição da tabela, como já vimos, é possível fazer isso:

```
create table Cliente (
...
DataCadastro datetime default (getdate()),
...
País varchar(20) default 'Brasil')
```

O valor de um default pode ser uma constante ou uma chamada função do sistema, como GETDATE(). Pode-se especificar ou não o nome da restrição, na forma CONSTRAINT *nome* logo antes da palavra "DEFAULT", por exemplo:

```
País varchar(20) constraint DefPais default 'Brasil'
```

Se o default for acrescentado com o comando ALTER TABLE, é preciso especificar para qual coluna ele vai ser ativado, por exemplo:

```
alter table Cliente
  add default 'Brasil' for País
```

Verificação [CHECK]

Uma restrição CHECK é muito semelhante a uma regra, que verifica os valores que estão sendo inseridos. A vantagem é que ele pode fazer referência a uma ou mais colunas da tabela. Por exemplo, vamos verificar, na tabela Cliente, se a Cidade e Estado são informados. Vamos criar uma restrição que impede de inserir o valor de Cidade, se Estado não foi informado:

```
alter table Cliente
  add check (not (Cidade is not null and Estado is null))
```

Para testar, tente inserir uma linha com Cidade = 'Teste' e Estado não informado (ou informado = NULL).

Note que a expressão do CHECK deve estar sempre entre parênteses. Sub-consultas não são permitidas em CHECK; para verificar dados em outras tabelas, use chaves estrangeiras como abaixo. Pode-se especificar ou não o nome da restrição, na forma CONSTRAINT *nome* logo antes das palavra "CHECK".

Tecnicamente, o que uma restrição CHECK faz é especificar uma condição de pesquisa Booleana (que retorna verdadeiro ou falso) que é aplicada a todos os valores inseridos para a coluna. Todos os valores que não retornem verdadeiro [TRUE] são rejeitados. Você pode especificar várias restrições CHECK para cada coluna.

Chave estrangeira [FOREIGN KEY]

Uma forma importante de integridade no banco de dados é a *integridade referencial*, que é a verificação de integridade feita entre duas tabelas. Por exemplo, a tabela 'ProdutoFornecedor' será usada para relacionar dados de produtos e fornecedores. Ela contém as colunas CodProduto e CodFornecedor. A primeira deve conter um código válido que exista na tabela 'Produto'. A outra deve ter um código válido existente na tabela 'Fornecedor'. Como validar essa verificação?

Uma *chave estrangeira* [foreign key] é uma restrição de integridade referencial. Ela consiste de uma coluna ou grupo de colunas cujo valor deve coincidir com valores de outra tabela. No nosso caso, vamos adicionar duas chaves estrangeiras na tabela 'ProdutoFornecedor': CodProduto faz referência a **Produto.CodProduto** e CodFornecedor faz referência a **Fornecedor.CodFornecedor**:

```
alter table ProdutoFornecedor
  add foreign key (CodProduto) references Produto(CodProduto),
    foreign key (CodFornecedor) references Fornecedor(CodFornecedor)
```

Note que a chave primária de Produto é (CodProduto) e de Fornecedor é (CodFornecedor), como foi definido antes. Se fossem especificados apenas os nomes das tabelas, sem indicar entre parênteses as colunas, também funcionaria:

```
alter table ProdutoFornecedor
   add foreign key (CodProduto) references Produto,
       foreign key (CodFornecedor) references Fornecedor
```

Porque nesse caso é assumida a chave primária. Mas uma chave estrangeira pode fazer referência a colunas que não a chave primária, desde que possuam uma restrição UNIQUE definida.

Outra forma de criar essas restrições é em conjunto com a tabela, da forma:

```
create table ProdutoFornecedor
(CodProduto int foreign key references Produto,
  CodFornecedor int foreign key references Fornecedor,
  primary key (CodProduto, CodFornecedor))
```

Esse tipo de restrição não cria um índice automaticamente, embora muitas vezes seja recomendável criar para maior desempenho (geralmente não-clustered). Pode-se especificar o nome da restrição opcionalmente, na forma CONSTRAINT *nome,* logo antes das palavras "FOREIGN KEY".

Gerenciando restrições com comandos SQL

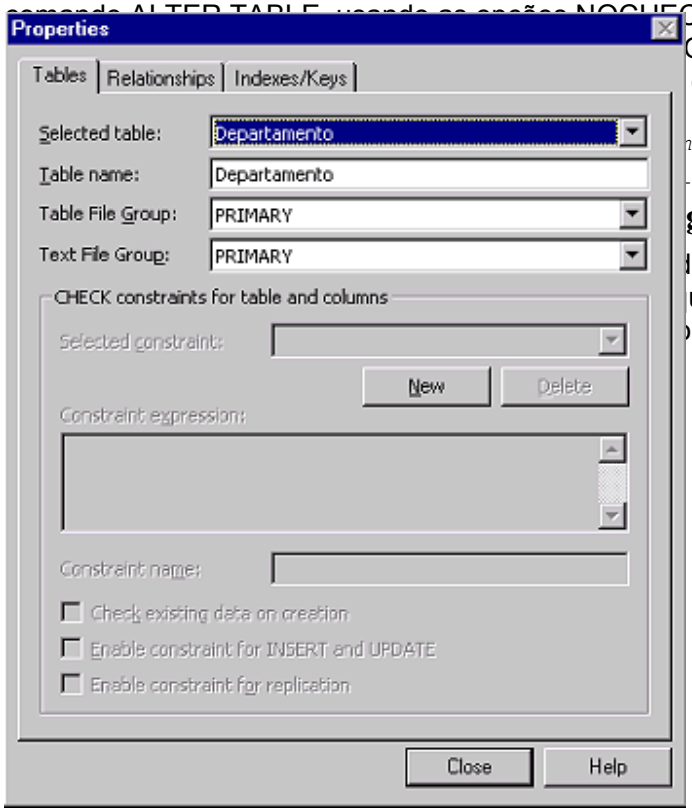
Como já vimos, CREATE TABLE pode criar as restrições junto com a tabela e ALTER TABLE, com a cláusula ADD, permite adicionar restrições depois que a tabela foi criada. Para excluir uma restrição, é preciso saber o seu nome. Se você não informou o nome na criação, terá que descobri-lo, o que pode ser feito usando-se:

```
sp_help nome_da_tabela
```

Esse comando mostra informações sobre a tabela, inclusive os nomes de cada restrição. Para excluir uma restrição, usa-se ALTER TABLE, com a opção DROP (independente do tipo de restrição). A sintaxe genérica é:

```
alter table nome_da_tabela
drop constraint nome_da_restrição
```

Uma restrição também pode ser desabilitada temporariamente e depois reabilitada com o comando ALTER TABLE, usando as opções NOCHECK (para desabilitar) e CHECK (para



QUE ou DEFAULT, apenas com as que fujam aos valores impostos pelas

```
ne_da_restrição
_da_restrição
```

ger

dem ser feitas através do Enterprise ue em **Design Table**, selecione alguma one Properties. Aparece a janela abaixo:

Você tem a página Indexes/Keys para criar ou remover a chave primária, e criar ou excluir restrições UNIQUE. A página "Tables" permite definir restrições CHECK, a página "Relationships" permite definir as chaves estrangeiras. Os defaults são tratados na lista de colunas, na janela de edição da tabela.
Note que nesta janela, você também pode definir em qual grupo de arquivos (ver grupos de arquivos) você vai criar cada uma das restrições.

11 - Visões,Gatilhos e Procedimentos

Visões [Views]

Procedimentos Armazenados

Gatilhos [Triggers]

Objetivos:
- Aprender a criar e utilizar visões e saber quais as particularidades do acesso a visões;
- Aprender a criar e utilizar procedimentos armazenados;
- Aprender a criar e utilizar triggers[gatilhos].

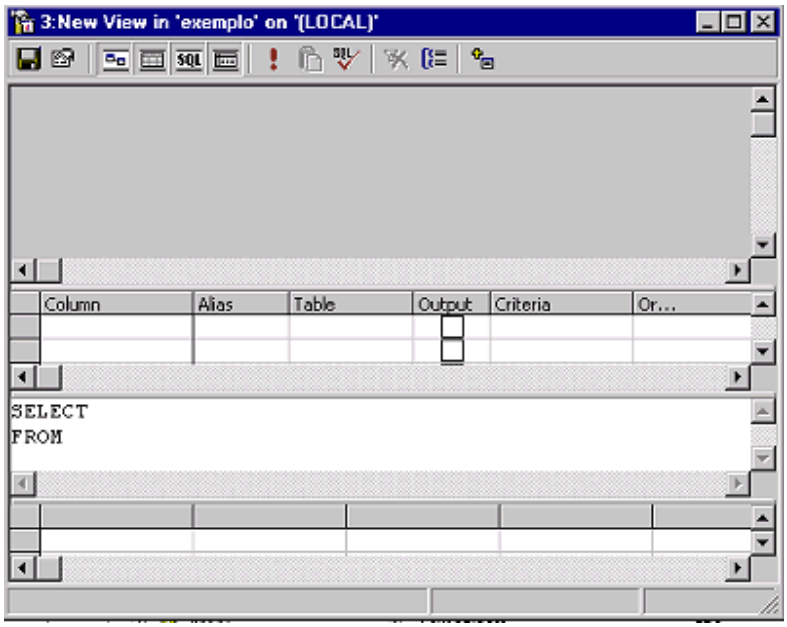
Visões [Views]

Uma visão [view] é uma forma alternativa de olhar os dados contidos em uma ou mais tabelas. Para definir uma visão, usa-se um comando SELECT que faz uma consulta sobre as tabelas. A visão aparece depois como se fosse uma tabela.
Visões têm as seguintes vantagens:

- Uma visão pode restringir quais as colunas da tabela que podem ser acessadas (para leitura ou para modificação), o que é útil no caso de controle de acesso, como veremos mais tarde.
- Uma consulta SELECT que é usada muito freqüentemente pode ser criada como visão. Com isso, a cada vez que ela é necessária, basta selecionar dados da visão.
- Visões podem conter valores calculados ou valores de resumo, o que simplifica a operação.
- Uma visão pode ser usada para exportar dados para outras aplicações.

Criando uma visão com o Enterprise Manager

Para criar uma visão com o Enterprise Manager, expanda um grupo de servidores, então o servidor em que está o baco de dados onde será criada a visão. Clique com o botão direito em **Views**. Aparece uma tela quase idêntica à do Query Designer. Caso alguma das seções aqui refrenciadas não esteja aparecendo, confira na seção em que tratamos do Query Designer, como ativá-la, e entenda melhor cada seção.



1. Na janela superior (logo abaixo dos ícones, chamada de seção do diagrama, clique com o botão direito, e selecione **Add Table**.
2. Na guia Tables (ou Views, caso você já tenha criado alguma visão e queira que ela faça parte desta que está sendo criada), selecione a tabela (ou visão) a ser adicionada, e então clique **Add**. Caso você queira remover alguma tabela adicionada ao diagrama, clique na mesma com~o botão direito e selecione **Remove**.
3. Repita tantas vezes quantas forem as tabelas (ou visões) a serem adicionadas à nova visão. Clique em **Close** quando tiver escolhido todas as tabelas (ou visões) desejadas.
4. Na caixa Column da seção da grade (parte da janela logo abaixo de onde estão as tabelas adicionadas), selecione as colunas a serem referenciadas na visão. Note que caso haja mais de uma tabela na seção do diagrama, quando você for selecionar a coluna na seção da grade, aparecerá o nome completo da coluna (tabela.coluna).
5. Marque a caixa **Output** se a coluna deve ser mostrada no resultado da visão. Note que você também pode escolher as colunas que farão parte da visão, selecionando-as na representação gráfica da tabela, mas as colunas selecionadas dessa maneira farão parte da saída por padrão. Para que não apareçam na saída, desmarque a caixa **Output**.
6. Para agrupar por alguma coluna, clique com o botão direito na coluna (na seção da grade) e selecione **Group By**.
7. Na coluna **Criteria**, digite o critério especificando quais linhas retornar; isso determina a cláusula WHERE. Se Group By for especificado, isso determina a cláusula HAVING.
8. Na coluna **Or...** entre com qualquer critério adicional para especificar quais linhas a serem retornadas.
9. Clique com o botão direito em qualquer lugar da seção da grade, e então selecione **Properties**.
    - "Output all columns" mostrará todas as linhas da visão no resultado.
    - "DISTINCT values" filtra os valores duplicados no resultado.
    - "Encrypt view" criptografa a definiçãp da visão.

- Opcionalmente, em "Top", entre com o número de linhas a serem retornadas no resultado. Digite a palavra PERCENT depois do número para mostrar uma porcentagem das linhas, no resultado.

10. Clique com o botão direito em qualquer lugar da seção do diagrama; clique então em **Run** (para ver o resultado) ou **Save** (para salvar a visão). Note que na seção SQL, aparece o código SQL do SELECT envolvido na criação da visão.

Criando uma visão com comandos SQL

Para criar uma visão através de SQL, use o comando CREATE VIEW. Esse comando tem a seguinte sintaxe:

```
CREATE VIEW nome_visão [(coluna [,...n])]
[WITH ENCRYPTION]
AS
    declaração_SELECT
[WITH CHECK OPTION]
```

nome_visão é o nome a ser dados à visão
coluna é o nome a ser usado para uma coluna em uma visão. Nomear uma coluna em CREATE VIEW só é necessário quando uma coluna é obtida por uma expressão aritmética, uma função, ou uma constante, ou quando duas ou mais colunas poderiam ter o mesmo nome (frequentemente por causa de uma junção), ou quando a coluna em uma visão recene um nome diferente do nome da coluna da qual se originou. Os nomes de colunas também podem ser atribuídos no comando SELECT. Caso você queira nomear mais de uma coluna, entre com o nome de cada uma separado por vírgulas.
WITH ENCRYPTION criptografa as entradas na tabela **syscomments** que contém o texto do comando CREATE VIEW.
WITH CHECK OPTION força todas as modificações de dados executadas na visão a aderirem aos critérios definidos na *declaração_SELECT*. Quando uma coluna é modificada através de uma visão, WITH CHECK OPTION garante que os dados permaneçam visíveis através da visão depois que as modificações forem efetivadas.
Vamos criar uma visão no banco de dados Exemplo, usando as tabelas 'Produto', 'Fornecedor' e 'ProdutoFornecedor'. Essa visão vai mostrar o nome do fornecedor e o nome do produto. Crie-a digitando o texto abaixo no Query Analyzer:

```
create view VisaoFornecProduto as
select f.Nome NomeFornecedor, p.Nome NomeProduto
from Fornecedor f
inner join ProdutoFornecedor pf
on f.CodFornecedor = pf.CodFornecedor
inner join Produto p
on pf.CodProduto = p.CodProduto
```

Para criar uma visão você deve estar posicionado no banco de dados onde a visão será criada ou então especificá-lo através da cláusula USES.
Ao criar uma visão, o texto do comando acima é armazenado na tabela *syscomments*. Agora, para testar, digite:

```
select * from VisaoFornecProduto
```

O resultado terá as colunas 'NomeFornecedor' e 'NomeProduto', mostrando os dados relacionados entre elas.
Você pode também criar uma visão que calcula valores usando colunas das tabelas, ou usando GROUP BY e funções agregadas, na declaração SELECT.

Alterando ou excluindo uma visão

Para alterar uma visão, você pode usar tanto o Enterprise Manager quanto o comando SQL, ALTER VIEW. Para alterá-la com o Enterprise Manager, selecione a visão que se quer alterar, clique na mesma com o botão direito e selecione **Design View**. Aparecerá a mesma janela vista na criação da visão com o Enterprise Manager, e aí você pode fazer as alterações que julgar necessárias à visão, salvar as alterações, executar a visão, etc.. Tudo da mesma forma que se você estivesse criando uma nova visão.
O comando SQL ALTER VIEW tem a seguinte sintaxe:

```
ALTER VIEW nome_visão [(coluna [,...n])]
   [WITH ENCRYPTION]
    AS declaração_select
  [WITH CHECK OPTION]
```

Todas as considerações feitas a respeito do comando CREATE VIEW se aplicam aqui. Caso você não se lembre do comando usado na criação da visão (o comando CREATE VIEW) , você pode obtê-lo usando o procedimento *sp_helptext*, da forma:

```
sp_helptext VisaoFornecProduto
```

Este texto é consultado na tabela *syscomments*. Algumas linhas podem aparecer quebradas no resultado.
É importante considerar que a alteração de uma visão não afeta os procedimentos armazenados ou gatilhos dependentes da mesma e não altera as permissões atribuídas à mesma (veremos mais sobre permissões em Segurança).

Modificando dados através de uma visão

Você pode executar um comando UPDATE em uma visão. Se ela foi baseada em uma única tabela, isso não provoca grandes problemas. Se a opção WITH CHECK OPTION acima for usada, as atualizações devem satisfazer as condições da cláusula WHERE usada na criação da visão. Inserções com INSERT também podem ser feitas.
Se a visão é baseada em duas ou mais tabelas, a atualização só é possível se o comando altera dados de apenas uma tabela. Colunas calculadas não podem ser alteradas. Se foram usadas funções de agregação, também não é possível modificar os dados através da visão. Na inserção, se uma coluna de uma tabela subjacente não permite nulos (NOT NULL), não é possível inserir linhas na visão, pois isso deixaria a coluna sem valor.

Procedimentos Armazenados

Um procedimento armazenado [stored procedures] é um conjunto de comandos SQL que são compilados e armazenados no servidor. Ele pode ser chamado a partir de um comando SQL qualquer.
Em versões anteriores do SQL Server, os procedimentos armazenados eram uma maneira de pré-compilar parcialmente um plano de execução. Quando da criação do procedimento armazenado, um plano de execução parcialmente compilado era armazenado em uma tabela de sistema. A execução de um procedimento armazenado era mais eficiente do que a execução de um comando SQL, porque o SQL Server não precisava compilar um plano de

excução completamente, apenas tinha que terminar a otimização do plano armazenado para o procedimento. Além disso, o plano de execução completamente compilado para o procedimento armazenado era mantido na cache de procedimentos do SQL Server, significando que execuções posteriores do porcedimento armazenado poderiam usar o plano de execução pré-compilado.
A versão 7.0 do SQL Server apresenta várias mudanças no processamento de comandos que estendem muitos dos benefícios de desempenho dos procedimentos armazenados para todos os comandos SQL. O SQL Server 7.0 não salva um plano parcialmente compilado para os procedimentos quando os mesmos são criados. Um procedimento armazenado é compilado em tempo de execução como qualquer outro comando Transact-SQL. O SQL Server 7.0 mantém planos de execução para todos os comandos SQL na cache de procedimentos, não apenas planos de execução de procedimentos armazenados. Ele então usa um algoritmo eficiente para comparação de novos comandos Transact-SQL com os comandos Transact-SQL de planos de execução existentes. Se o SQL Server 7.0 determinar que um novo comando Transact-SQL é o mesmo que um comando Transact-SQL de um plano de execução existente, ele reutiliza o plano. Isso reduz o ganho relativo de desempenho, na pré-compilação de procedimentos armazenados, já que estende a reutilização de planos de execução para todos os comandos SQL.
A vantagem de usar procedimentos armazenados é que eles podem encapsular rotinas de uso freqüente no próprio servidor, e estarão disponíveis para todas as aplicações. Parte da lógica do sistema pode ser armazenada no próprio banco de dados, em vez de ser codificada várias vezes em cada aplicação.

Criando procedimentos armazenados

Para criar um procedimento, use o comando CREATE PROCEDURE. Por exemplo, o procedimento abaixo recebe um parâmetro (@nome) e mostra todos os clientes cujo nome contenha o nome informado:

```
create procedure BuscaCliente
@nomeBusca varchar(50)
as
select CodCliente, Nome from Cliente
where Nome like '%' + @nomeBusca + '%'
```

Note que os parâmetros são sempre declarados com @, logo após o nome do procedimento. Um procedimento pode ter zero ou mais parâmetros. Declara-se o nome do procedimento, e a seguir o tipo de dados do parâmetro.
Nota: ao invés de CREATE PROCEDURE, pode-se utilizar CREATE PROC, com o mesmo efeito.
Dentro do procedimento pode haver vários comandos SELECT e o resultado desses comandos será o resultado do procedimento. O corpo do procedimento começa com a palavra AS e vai até o final do procedimento.
Não se pode usar os comandos SET SHOWPLAN_TEXT, e SET SHOWPLAN_ALL dentro de um procedimento armazenado, pois os mesmos devem ser os únicos comandos de um lote (batch).
Dentro de um procedimento, nomes de objetos usados em alguns comandos devem ser qualificados com o nome do propietário do objeto, se outros usuários utilizarão o procedimento armazenado. Os comandos são:

- ALTER TABLE
- CREATE INDEX
- Todos os comandos DBCC
- DROP TABLE

- DROP INDEX
- TRUNCATE TABLE
- UPDATE STATISTICS

Executando procedimentos armazenados

Para executar um procedimento, usa-se o comando EXEC (ou EXECUTE). A palavra "EXEC" pode ser omitida se a chamada de procedimento for o primeiro comando em um script ou vier logo após um marcador de fim de lote (a palavra "GO").
Por exemplo, execute o procedimento anterior da seguinte forma:

```
BuscaCliente 'an'
```

O resultado será as linhas da tabela Cliente onde o valor de Nome contém 'an' (se existirem tais linhas).
Ao executar um procedimento, você pode informar explicitamente o nome de cada parâmetro, por exemplo:

```
BuscaCliente @nomeBusca = 'an'
```

Isso permite passar os parâmetros (se mais de um) fora da ordem em que eles foram definidos no procedimento.
EXEC também pode executar um procedimento em outro servidor. Para isso, a sintaxe básica é:

```
EXEC nome_servidor.nome_banco_de_dados..nome_procedimento
```

Comandos para uso em procedimentos armazenados

Você pode declarar uma variável em um procedimento e usá-la para guardar valores. Por exemplo, exclua o procedimento anterior e crie-o novamente como abaixo:

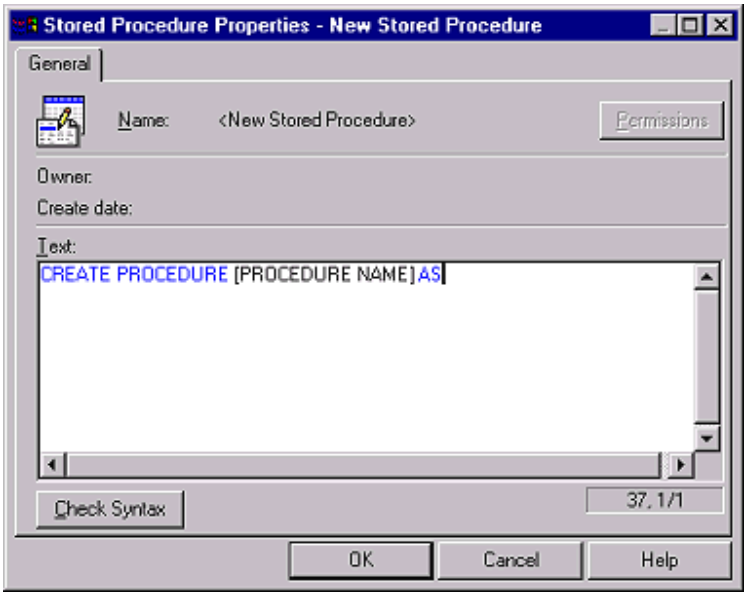
```
drop procedure BuscaCliente
go
create procedure BuscaCliente
  @nomeBusca varchar(50)
as
    declare @contagem int, @mensagem char(100)
    select CodCliente, Nome from Cliente
    where Nome like '%' + @nomeBusca + '%'
   -- conta quantas linhas foram encontradas
    select @contagem = count(*) from Cliente
    where Nome like '%' + @nomeBusca + '%'

    if @contagem = 0
    begin
      select @mensagem = 'Nenhum cliente contém
"'+@nomeBusca+'"'
      print @mensagem
      print ""
    end
```

O comando DECLARE declara variáveis, que são sempre introduzidas pelo caractere @. No caso, @contagem é uma variável do tipo int e @mensagem do tipo char(100).
Note que quando você usa um comando SELECT, o resultado pode ser colocado numa variável, como @contagem acima. Esse resultado não aparece no resultado do SELECT. Essa é também a única forma de alterar uma variável (você não pode escrever '@variável = valor' diretamente).
O comando IF verifica uma condição e executa um comando caso a condição seja verdadeira. Se acompanhado da cláusula ELSE, executa um outro comando caso a condição seja falsa. O comando PRINT usado acima é geralmente usado para mostrar mensagens, que aparecem quando você chama o procedimento interativamente.

Os comandos BEGIN e END são usados para delimitar uma lista de comandos, que passa a ser tratada como um comando único. No caso acima, eles são necessários para poder executar três comandos dentro do IF (o SELECT e os dois PRINT).

Criando procedimentos armazenados com o Enterprise Manager



Também é possível a criação de procedimentos armazenados através do Enterprise Manger. Para isso, deve-se expandir um grupo de servidore, um servidor, e o banco de dados onde o procedimento armazenado será criado. Clique então com o botão direito em **Stored Procedures**, e selecione **New Stored Procedure**. Aparece uma tela como abaixo

Nessa tela você deve dar o nome que desejar ao procedimento, substituindo o texto em preto [PROCEDURE NAME] pelo nome que você quer dar ao procedimento armazenado sendo criado.
Logo depois do AS, você deve entrar com o código do procedimento armazenado, conforme descrito acima. Você pode após entrar com o código desejado, clicar no botão **Check Syntax**, que verificará se há erros de sintaxe nas declarações SQL.
Quando tiver terminado de entrar com o código do procedimento, basta clicar em OK que o mesmo será criado.

Gatilhos [Triggers]

Um *gatilho* [trigger] é um tipo de procedimento armazenado, que é executado automaticamente quando ocorre algum tipo de alteração numa tabela. Gatilhos "disparam" quando ocorre uma operação INSERT, UPDATE ou DELETE numa tabela.
Geralmente gatilhos são usados para reforçar restrições de integridade que não podem ser tratadas pelos recursos mais simples, como regras, defaults, restrições, a opção NOT NULL etc. Deve-se usar defaults e restrições quando eles fornecem toda a funcionalidade necessária.

Um gatilho também pode ser usado para calcular e armazenar valores automaticamente em outra tabela, como veremos.

Exemplo de gatilhos

Para utilizar gatilhos, vamos criar antes algumas tabelas que serão usadas como exemplo. A tabela "NotaFiscal" conterá os cabeçalhos de notas fiscais. A tabela "ItemNotaFiscal" irá conter itens de nota fiscal relacionados com as notas fiscais. Execute o script abaixo para criar as tabelas:

```
create table NotaFiscal
   (NumeroNota numeric(10) primary key,
   ValorTotal numeric(10,2) default (0) )
GO
create table ItemNotaFiscal
   (NumeroNota numeric(10) foreign key references NotaFiscal,
     CodProduto int foreign key references Produto,
     Quantidade int not null check (Quantidade > 0),
     primary key (NumeroNota,CodProduto)
    )
```

Vamos usar gatilhos para duas finalidades: primeiro, quando for excluída uma nota fiscal, todos os seus itens serão excluídos automaticamente. Depois, quando for incluído um item, a coluna 'ValorTotal' será atualizada, na tabela 'NotaFiscal'.

Criando gatilhos

Gatilhos são sempre criados vinculados a uma determinada tabela. Se a tabela for excluída, todos os gatilhos dela são excluídos como conseqüência. Ao criar um gatilho, você pode especificar qual(is) a(s) operação(ões) em que ele será acionado: INSERT, UPDATE ou DELETE.

Gatilhos para inserção

Quando é feita a inclusão de uma ou mais linhas na tabela, o SQL Server cria uma tabela virtual chamada *inserted*, que contém as linhas que serão incluídas (mas ainda não foram). Essa tabela tem a mesma estrutura da tabela principal. Você pode consultar dados nessa tabela com o SELECT, da mesma forma que uma tabela real.
Vamos criar um gatilho, chamado InclusaoItemNota, que será ativado por uma operação INSERT na tabela ItemNotaFiscal. Primeiro ele vai verificar se os valores sendo inseridos possuem uma NotaFiscal relacionada ou não. Digite o seguinte comando:

```
create trigger InclusaoItemNota
on ItemNotaFiscal for insert
as
   if not exists (select * from
inserted, NotaFiscal
where inserted.NumeroNota =
NotaFiscal.NumeroNota)
raiserror('Esse item não contém um número de nota válido')
     update NotaFiscal
     set ValorTotal = ValorTotal
         + (select i.Quantidade * p.Preço
              from Produto p, inserted i
             where p.CodProduto = i.CodProduto)
               where NumeroNota = (select NumeroNota from inserted)
```

Primeiro o gatilho usa as tabelas *inserted* e *NotaFiscal* para consultar se existe o valor de NumeroNota na tabela. Caso não exista, o comando RAISERROR gera um erro de execução, com uma mensagem que será retornada para a aplicação. Esse comando efetivamente *cancela* o comando INSERT que estiver sendo executado.

Depois verifica a quantidade que está sendo inserida (inserted.Quantidade) e multiplica pelo preço do produto (Produto.Preço). Esse preço é buscado na tabela de produtos, usando como valor de pesquisa o código do produto inserido (inserted.CodProduto). Ele atualiza a nota fiscal relacionada com o item que está sendo inserido (para isso verifica where NumeroNota=(select NumeroNota from inserted)).

Gatilhos para exclusão

Na exclusão, as linhas da tabela são removidas e colocadas na tabela virtual *deleted*, que tem a mesma estrutura da tabela principal. Um gatilho para exclusão pode consultar *deleted* para saber quais as linhas excluídas.
Vamos criar um gatilho, na tabela NotaFiscal para, quando a nota fiscal for excluída, todos os seus itens de nota relacionados, na tabela ItemNotaFiscal, sejam excluídos em cascata.
Execute o seguinte:

```
create trigger ExclusaoNota
on NotaFiscal for delete
as
   -- excluir todos os itens relacionados
   -- (mesmo NumeroNota que deleted)
   delete from ItemNotaFiscal
   where NumeroNota in (select NumeroNota from deleted)
```

Gatilhos para atualização

As tabelas *inserted* e *deleted*, como já vimos, são tabelas virtuais que podem ser usadas dentro de um gatilho. A primeira contém os dados que estão sendo inseridos na tabela real e a segunda contém os dados antigos, que estão sendo incluídos.
Num gatilho de atualização (FOR UPDATE), essas duas tabelas também estão disponíveis. No caso, *deleted* permite acessar os dados como eram *antes* da modificação e *inserted* permite acessar os dados *depois* da atualização.
Podemos então considerar uma atualização como uma exclusão seguida de uma inserção (excluem-se valores antigos e inserem-se valroes novos).
Por exemplo, ao mudar a Quantidade em um ItemNotaFiscal, o total da nota deve ser recalculado. Para isso, é preciso levar em conta a diferença entre a quantidade antiga (deleted.Quantidade) e a nova (inserted.Quantidade). Vamos criar um gatilho em ItemNotaFiscal que faz isso:

```
create trigger AlteracaoItemNota
on ItemNotaFiscal for update
as
    if update(Quantidade) or update(CodProduto)
    begin
        update NotaFiscal
        set ValorTotal = ValorTotal +
        (select p.Preço * (i.Quantidade - d.Quantidade)
          from Produto p inner join inserted i
            on p.CodProduto = i.CodProduto inner join deleted d
            on i.CodProduto = d.CodProduto and i.NumeroNota = d.NumeroNota)
    end
```

Note acima o uso de 'if update(*nome_da_coluna*)'. Dentro de um gatilho de atualização, isso permite descobrir se a coluna está sendo alterada ou não. Isso para evitar trabalho desnecessário se não estiver sendo modificada uma dessas colunas.

Criando gatilhos para múltiplas ações

Um gatilho pode ser criado para uma tabela para múltiplas operações nessa tabela. Por exemplo, para criar um gatilho usado em INSERT, UPDATE e DELETE, usa-se uma sintaxe, como:

```
create trigger nome_do_gatilho
on nome_da_tabela for INSERT, UPDATE, DELETE
as
texto_do_gatilho
```

Outros comandos

Em gatilhos, assim como em procedimentos armazenados, é possível declarar variáveis e usar comandos como IF, BEGIN..END etc.
Alguns comandos não são permitidos dentro de um gatilho. Estes são:

```
ALTER DATABASE            ALTER PROCEDURE   ALTER TABLE
ALTER TRIGGER             ALTER VIEW              CREATE DATABASE
CREATE DEFAULT            CREATE INDEX            CREATE PROCEDURE
CREATE RULE               CREATE SCHEMA           CREATE TABLE
CREATE TRIGGER            CREATE VIEW             DENY
DISK INIT                 DISK RESIZE             DROP DATABASE
DROP DEFAULT              DROP INDEX              DROP PROCEDURE
DROP RULE                 DROP TABLE              DROP TRIGGER
DROP VIEW                 GRANT                   LOAD DATABASE
LOAD LOG                  RESTORE DATABASE  RESTORE LOG
REVOKE                    RECONFIGURE             TRUNCATE TABLE
UPDATE STATISTICS
```

12 - 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, a 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.

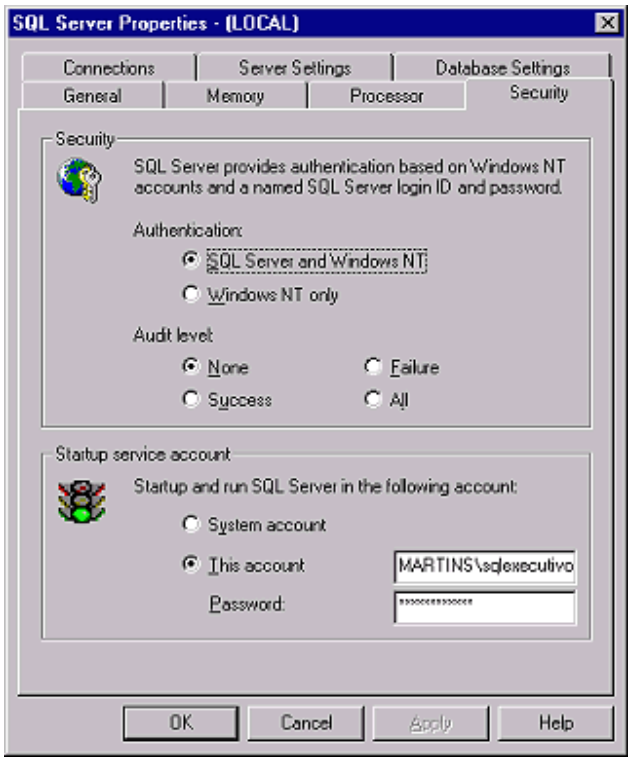
Quando o SQL Server lhe pede um login e senha, ele usa seu próprio cadastro de usuários, independente do banco de dados de contas do Windows NT. Os logins de usuário devem ser cadastrados no SQL Server.
Modo de autenticação de segurança do Windows NT [Windows NT Server Authtentication Security Mode]: 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 atreas administrativas a outras pessoas. Por exemplo, você poderia querer se dsignar como o DBA senior, com a habilidade de excutar 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 conhceidos como papéis fixos de servidor).
O SQL Server 7.0 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 sobrepôr-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 prdefinidos de bancos de addos 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 namcos de dados, e sobrepôr-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 semlhante 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 papél 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 obtem o mesmo efeitos 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.

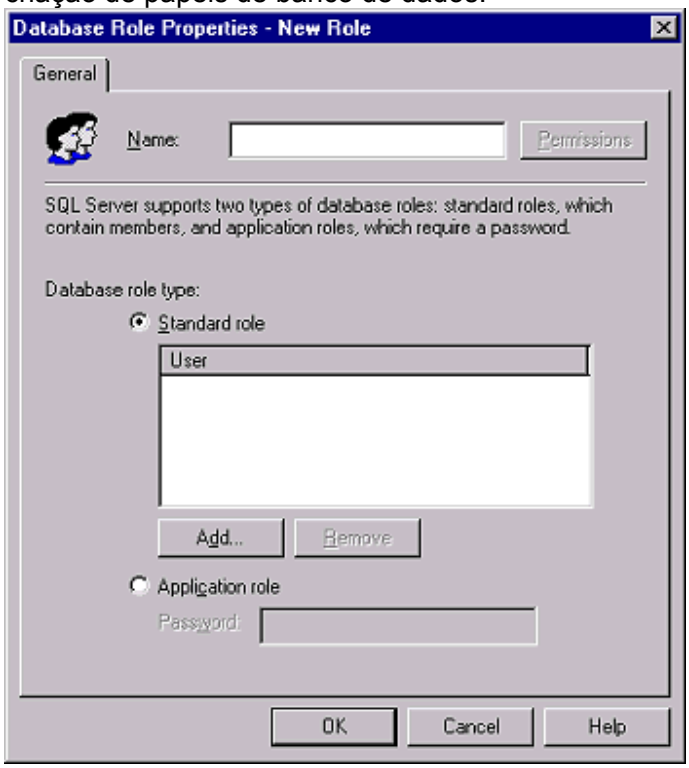
Quando for criar papéis personalizados de banco de dados, tenha o seguinte em mente:

- Papéis personalizados de banco de dados, como ocorre com qualqeur 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

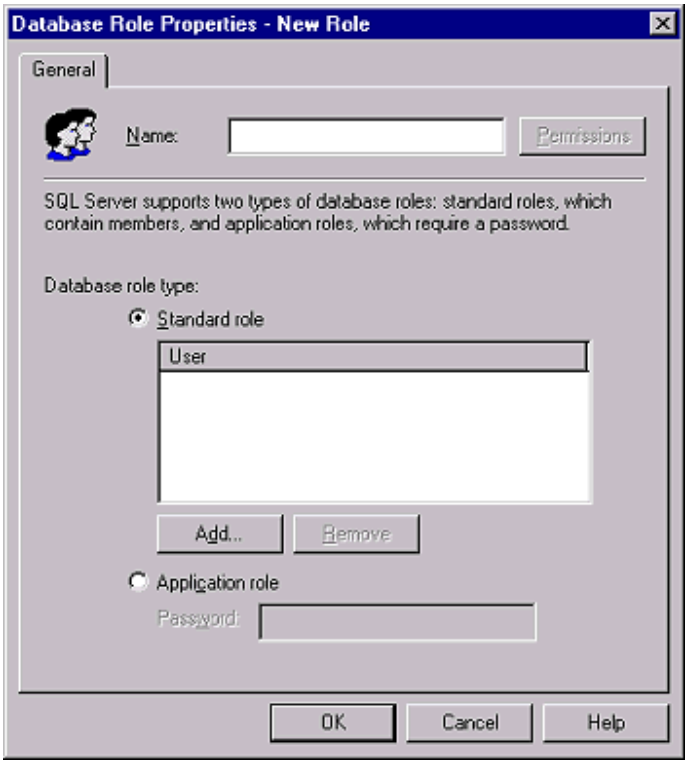
Como parte da sua responsabilidade cotidiana de manter o SQL Server, você achará necessário às vezes remover IDs de usuário de bancos de dados e, com menor freqüência, remover papéis de banco de dados que não sejam mais necessários. A remoção de IDs de usuário de banco de dados será vista em Como excluir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

1. No Enterprise Manager, expanda o banco de dados em que está definido o papel que você quer excluir.
2. 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.
3. Caso haja algum usuário, remova-o selecionando-o e clicando no botão **Remove**.
4. Feito isso, clique em Ok, selecione o papel a ser excluído, clique no mesmo com o botão direito e selecione **Delete**.
5. 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 loginm 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:

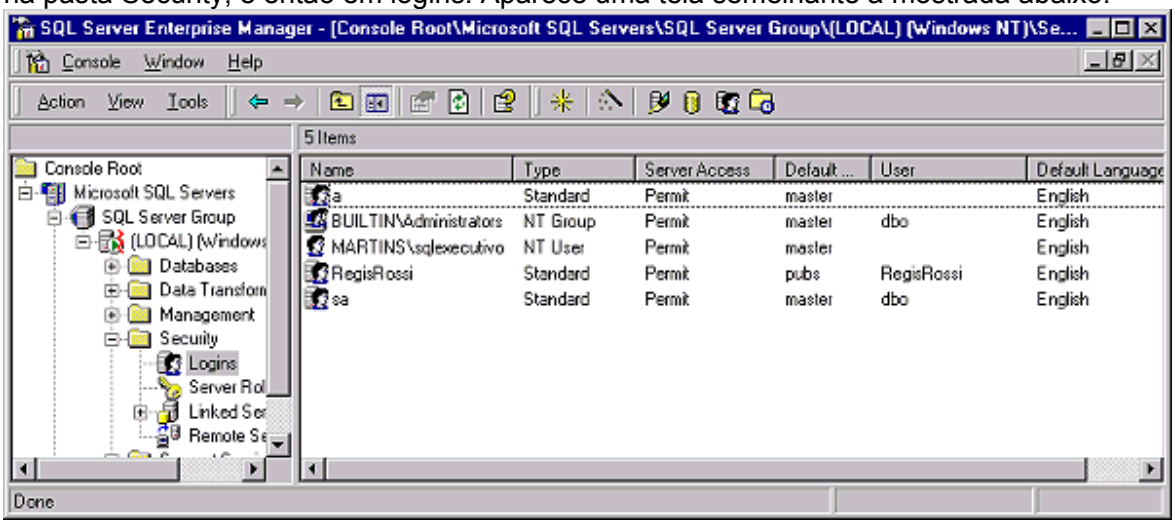
- Para adicionar um login ao papel de servidor, clique no botão **Add**. Aparece a caixa de diálogo "Adicionar Membros" [Add Mebers], com todos os logins definidos para o servidor.
- Escolha um ou mais logins para adicionar a esse papel de servidor. Cada vez que você selecionar um login, ele ficará marcado, 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

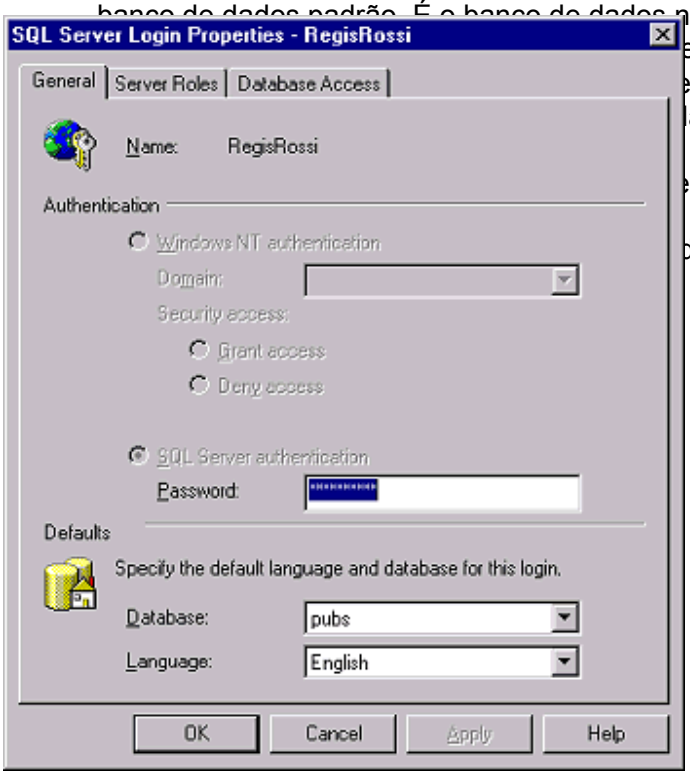
Visualizando informações de logins do SQL Server

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



eira vez.

e este usuário recebeu no banco de dados

la primeira vez que efetua logon no SQL

específica para o login. O padrão é inglês

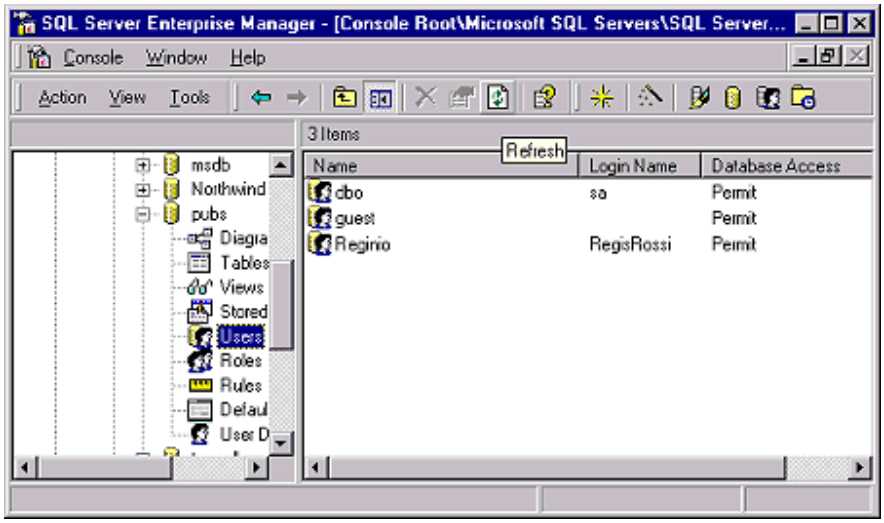
dos logins, clique no mesmo com o botão

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.

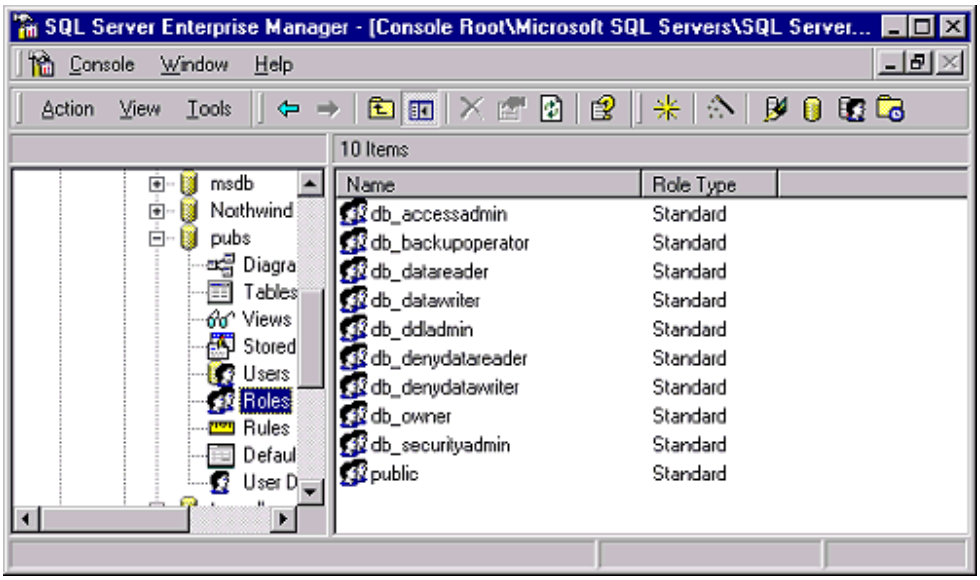
- A coluna "Database Acces" indica o tipo de acesso que o ID de usuário tem a esse banco de dados.

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.

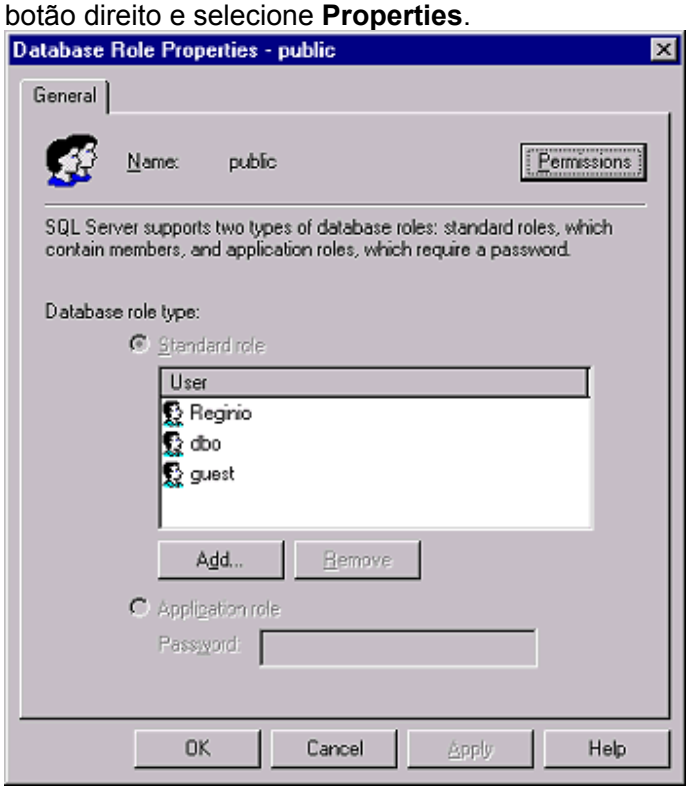
Visualizando informações de papéis de bancos de dados.

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**.

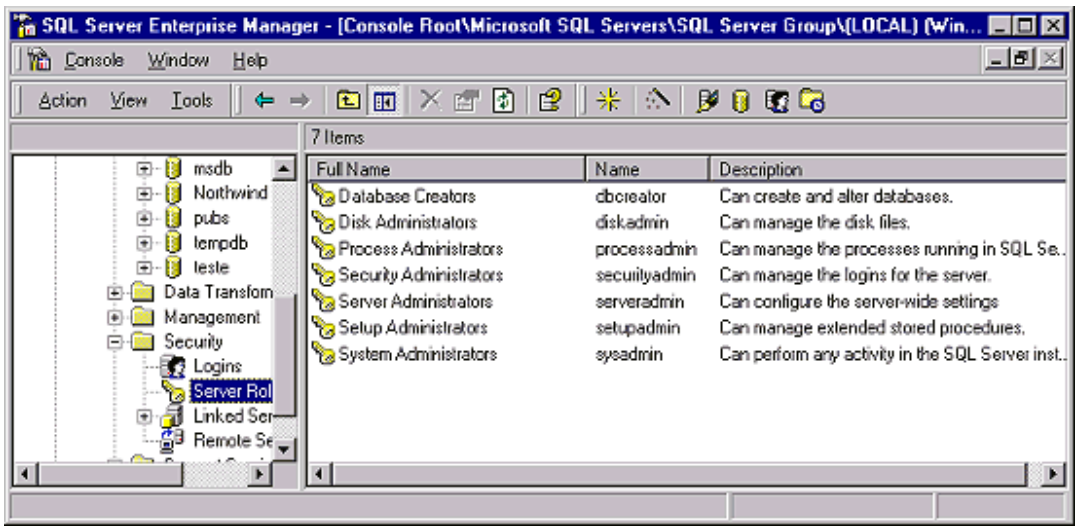


Essa caixa de diálogo lista os usuários do banco de dados que fazem parte deste papel em particular.
Para sair dessa caixa de diálogo, clique em Cancel.
É bem provável que você ache mais fácil ver estas informações através das informações de login, como descrito anteriormente.

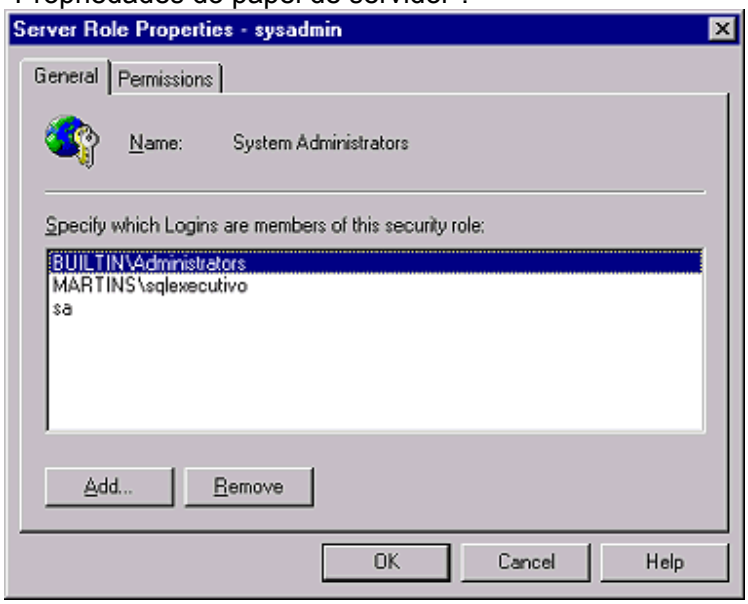
Visualizando informações de papéis de servidor

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 realziar 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 obtem 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

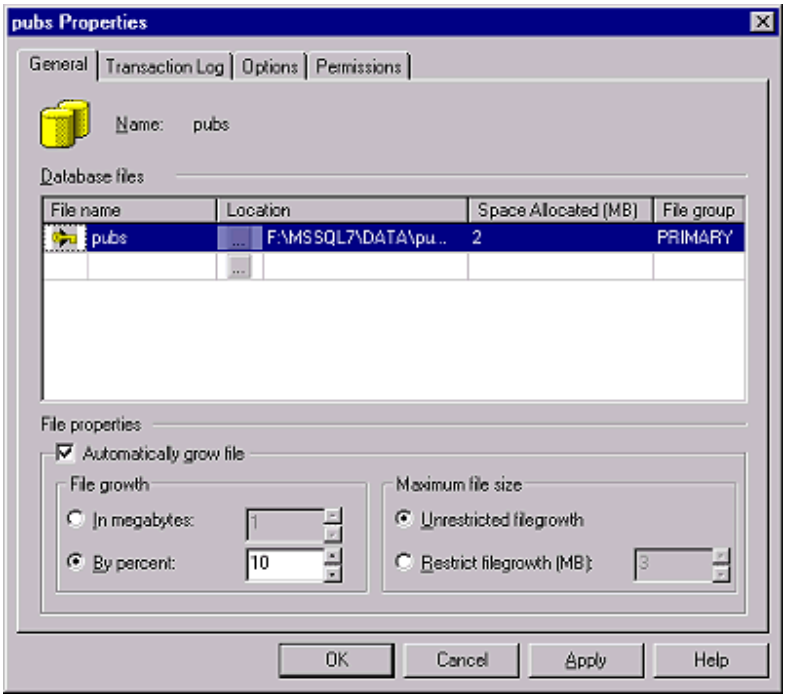
Cinco níveis de permissões podme ser atribuídas a um usuário, conforme segue:

- 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

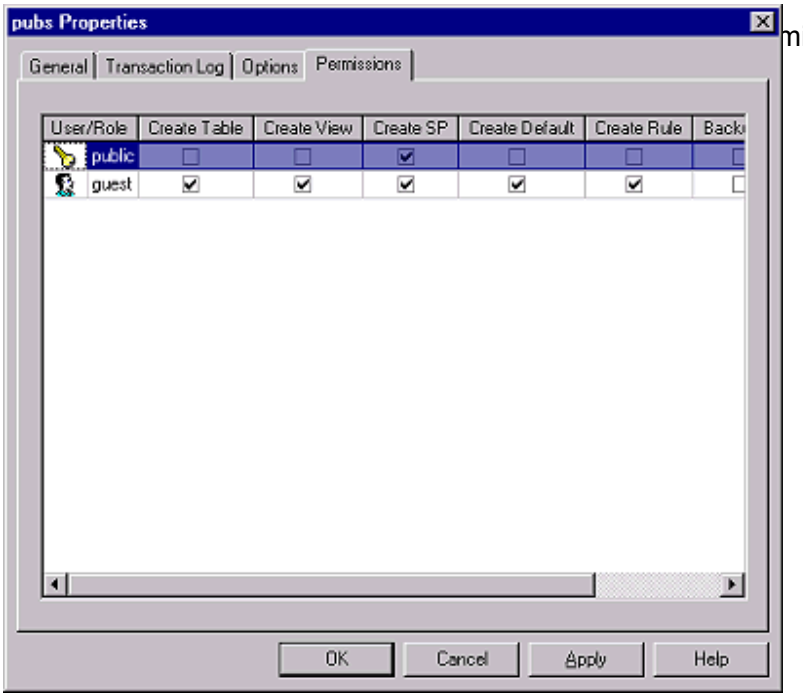
Antes que você aprenda a conceder e revogar permissões para usuários, grupos ou papéis, é importante que você saiba como visualizar permissões tanto de objetos como de comandos.



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 addos, 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:



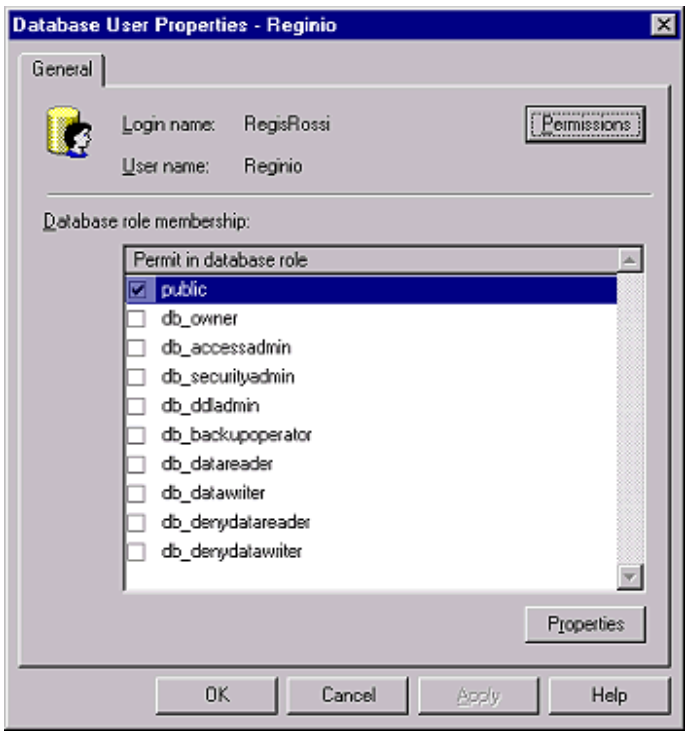
missões para comandos SQL

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.

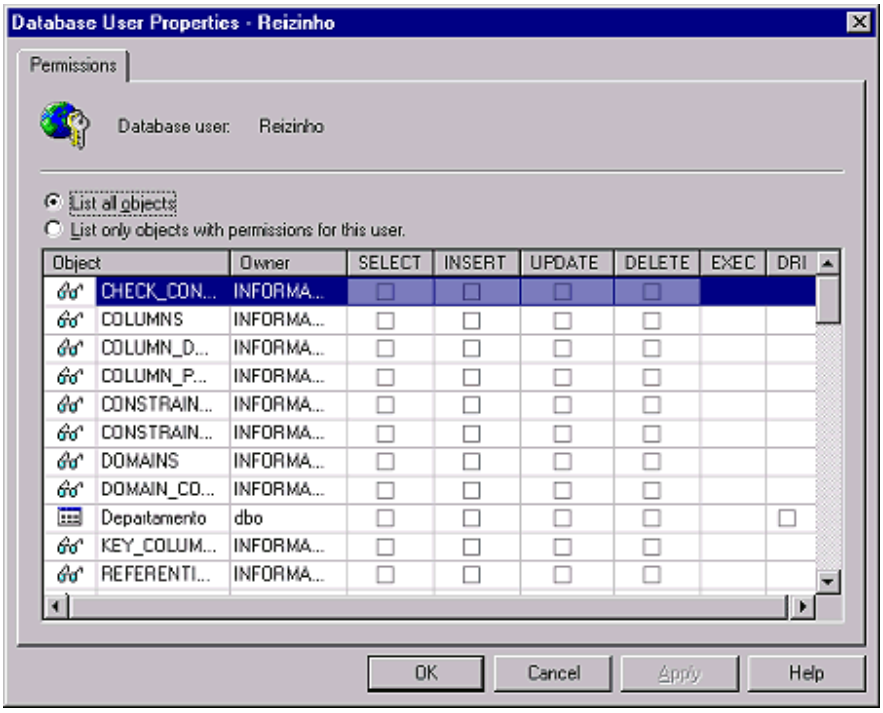
Visualizando permissões de objetos

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:

1. No Enterprise Manager, expanda o banco de dados cujas permissões de objetos você quer visualizar.
2. 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.
3. 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).



4. Clique no botão Permissions para ver as permissões a nível de objeto que esse usuário



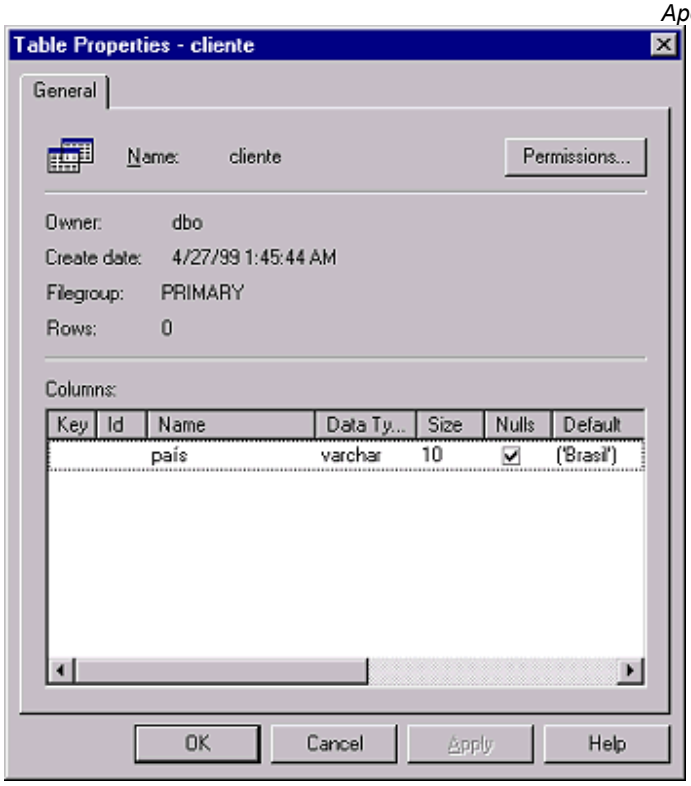
(ou papel) tem.

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**.

5. 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)

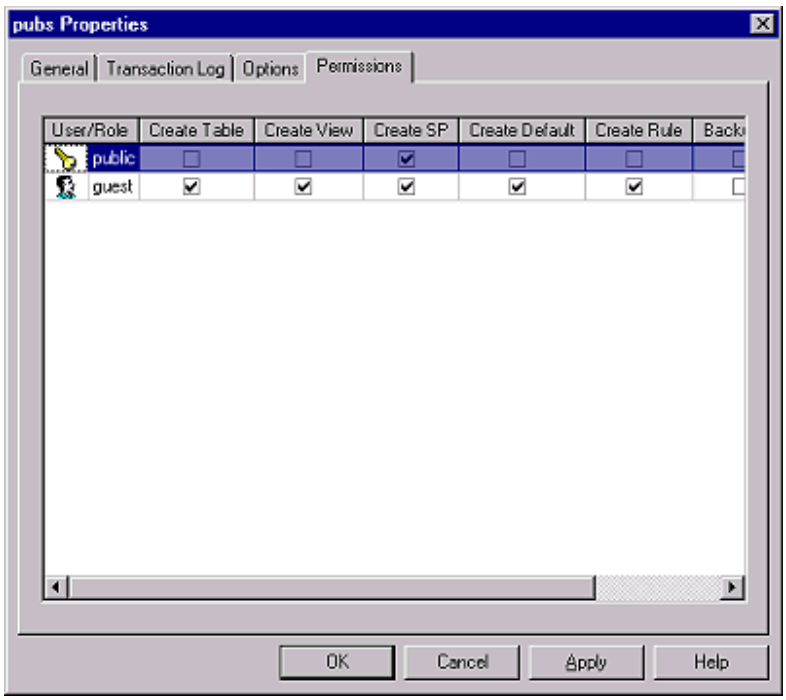
4. Para exibir as permissões para esse objeto, clique no botão Permissions. Aparece a tela de permissões do objeto, como abaixo:

   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

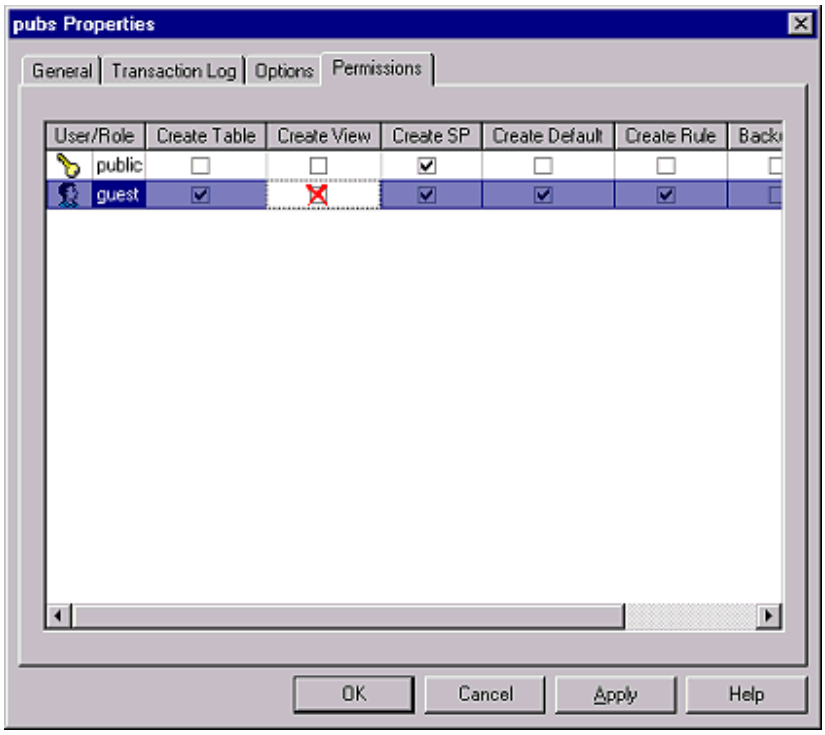
A concessão e revogação de permissões usa as mesmas telas que acabamos de ver. Mas agora, atribuíremos e revogaremos permissões de grupos, usuários, e papéis.
Lembre-se que nenhum usuário tem qualquer permissão para acessar qualquer objeto de dados até que você explicitamente atribua a ele tais permissões. 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 paara 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.

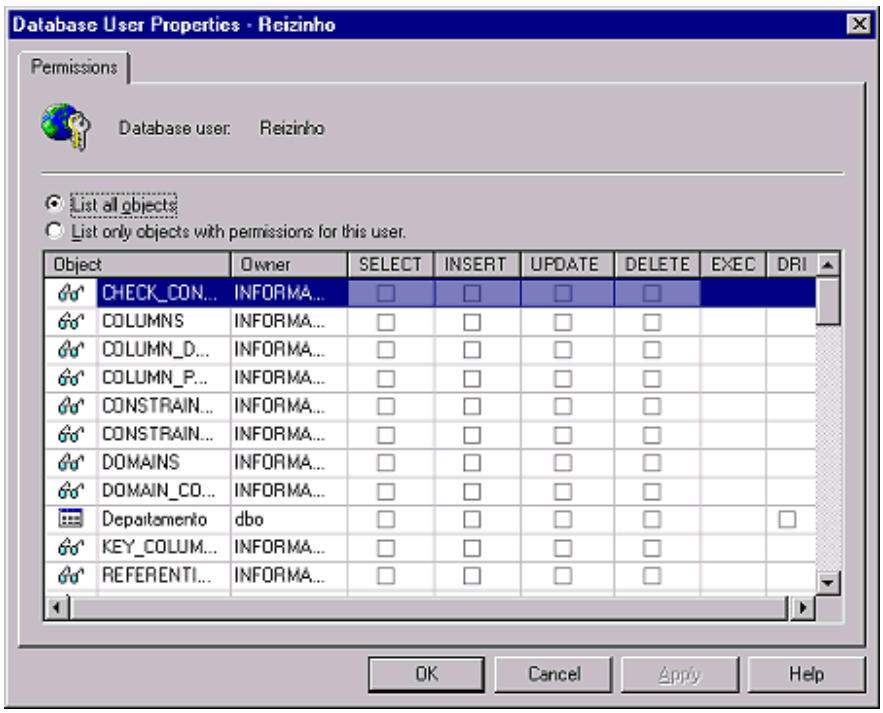
5. Depois de revogar e conceder todas as permissões para comandos SQL que você queira, saia dessa tela clicando em Ok. Então você volta para o Enterprise Manager.

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
```

13 - Backup e Restauração

Dispositivos de Backup

Implementar Backup

Restaurar um Backup

Agendar Backups Automáticos

Objetivos:

- Aprender a gerenciar os dispositivos de backup;
- Aprender como fazer backups, restaurar os dados, e agendar backups automáticos.

Conceitos

Um *backup* ou *dump* do banco de dados é a operação de copiar os dados para um dispositivo de backup. Pode ser feito com o Enterprise Manager ou com o comando BACKUP. Não é necessário parar o SQL Server ou desconectar os usuários para fazer a operação de backup. Ela pode ser feita a qualquer momento. Deve-se considerar que a realização do backup com usuários utilizando o banco de dados, causa uma pequena queda de performance, que pode ser perceptível aos usuários. É importante então escolher horas de menor atividade do servidor (ou ao menos do banco de dados cujo backup está sendo feito) para a realização do backup.
Uma *restauração* ou *RESTORE* do banco de dados é a operação de trazer os dados de um meio de backup de volta para os bancos de dados.

Tipos de backup

Um backup pode ser feito do banco de dados inteiro, que copia todos os dados, mais o log de transações (a tabela *syslogs*). Se esse backup for restaurado, todo o conteúdo do banco de dados é restaurado e sobrescreve o conteúdo atual. Esse tipo de backup pode ser feito com o comando BACKUP DATABASE ou o Enterprise Manager.
Pode ser feito um backup apenas do log de transações. Como o log de transações contém apenas as modificações feitas aos dados, se esse backup for restaurado, apenas essas modificações serão aplicadas sobre os dados. Esse tipo de backup pode ser feito com o comando BACKUP LOG ou o Enterprise Manager. Após um backup desse tipo, o log de transações é esvaziado (exceto as transações que ainda estão sendo atualizadas). Um backup do log de transações leva muito menos tempo para ser feito do que um bakcup de todo banco de dados. Assim, em um banco de dados que é bastante modificadao diariamente, pode-se fazer diversos backups diários do log de transações. Ou então algum backup diferencial, como será mostrado a seguir.
Pode-se fazer também o que é chamado de *backup diferencial*. Esse tipo de backup é semelhante ao backup do log de transações, com a diferença de que só fará backup dos valores modificados desde o último backup completo (de todo o banco de dados). Ou seja, se uma infromação foi modificada vinte vezes desde o último backup, um backup do log de transações teria as 20 modificações feitas nessa informação, enquanto que um backup diferencial teria apenas o último valor armazenado. Assim, esse tipo de backup gera arquivos menores, apesar de demorar um pouco mais de tempo para ser realizado que o backup do log de transações. Mas, por outro lado, exige bem menos tempo para restauração.
Por último, existe a possibilidade de se fazer backup de arquivos individuais do banco de dados. Lembre-se que um banco de dados pode ser formado por vários arquivos. Assim, para um banco de dados muito grande, a ponto de não poder ser "backupeado" em uma única noite, por exemplo, podem ser feitos backups dos arquivos que o formam, um por vez.
Nota: os comandos DUMP DATABASE e DUMP TRANSACTION, existentes em versões do SQL Server anteriores à 7.0, ainda existem, mas apenas por motivos de compatibilidade. Recomenda-se utilizar BACKUP DATABASE, ao invés de DUMP DATABASE, e BACKUP LOG ao invés de DUMP TRNSACTION, já que DUMP não será mais aceito em futuras versões do SQL Server.

Dispositivos de Backup

Um dispositivo de backup nada mais é que um ponteiro para o local onde o backup do seu banco de dados será armazenado. Pode-se criar um dispositivo de backup a qualquer instante,

no Enterprise Manager, ou criá-lo apenas quando se for de fato fazer um backup do banco de dados. Um dispositivo de bakcup opde especificar o nome de um arquivo em disco rígido que será escrito quando o bakcup for executado, ou pode especificar o nome de uma unidade de fita.
A criação de um dispositivo de backup **não** cria um arquivo até que seja executado o backup..

Gerenciando dispositivos de backup

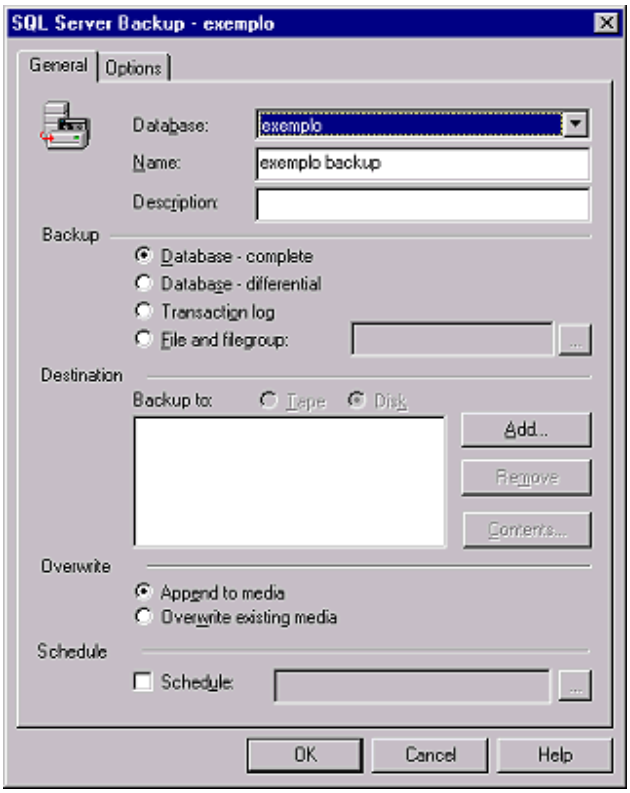
Para criar um dispositivo de backup, faça o seguinte:

- No Enterprise Manager, selecione o servidor do qual se quer fazer backup de algum banco de dados, abra a pasta **Management** e clique com o botão direito em Backup. Selecione **New Backup Device**. Cada dispositivo de backup tem um nome lógico e um nome físico.
    - Em Name, informe um "apelido" (nome lógico) que será utilizado pelo comando de backup quando da realização de um backup do banco de dados. No nosso caso, informemos Backup1.
    - O campo **Tape Drive Name** ou **File Name** é a localização física do arquivo em que o dispositivo de backup irá escrever.
- Para fitas, informe o nome da unidade de fita, que é algo como \\.\TAPE0. Para arquivos em disco, informe o nome e caminho do arquivo de disco, algo como \\Servidor\backups\Segunda\master.bak
- Clique em Ok para terminar. Aparece agora o nome deste dispositivo de backup do lado direito da tela, quando se seleciona Backups do lado esquerdo do Enterprise Manager.

O Enterprise Manager cria o nome físico do arquivo em *C:**MSSQL*\BACKUP, como default, mas pode ser mudado. Se você clica em "Tape backup device", você deve escolher um dos dispositivos de fita suportados pelo Windows NT, como nome físico. No caso, usaremos "Disk backup device". Clique em Ok.
Note que o arquivo, no caso *c:**mssql*\backup\Backup1.dat, não é criado imediatamente, só da primeira vez que um backup for feito. Ele também expande automaticamente, dependendo dos dados colocados nele, por isso você não define o seu tamanho.
Nota: O suporte a fitas do SQL Server depende do suporte fornecido pelo Windows NT (ver Painel de Controle, ícone "Dispositivos de fita" [Tape Devices]).

Implementando um backup

Para fazer um backup *completo* do banco de dados Exemplo, faça o seguinte:

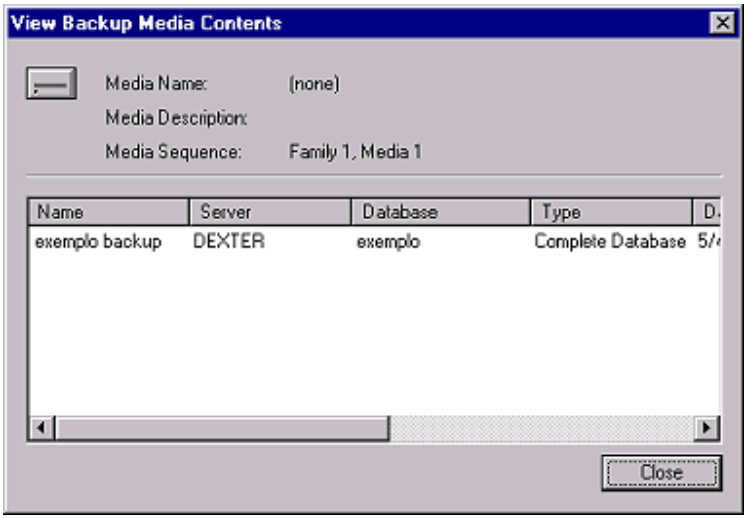


1. no Enterprise Manager, localize esse banco de dados sob "Databases" e clique nele com o botão direito. Selecione **All Tasks**, e **Backup Database**. Aparece a janela abaixo:

2. Informe um nome para o backup (ou deixe o nome padrão, que é *nome_banco_de_ dados backup*),e caso queira, uma descrição [Description] para o backup. Você pode escolher entre quatro opções de backup:
   - backup do banco de dados inteiro (Database - complete): a opção selecionada por padrão.
   - backup diferencial (Database - differential): um backup diferencial faz uma cópia apenas das mudanças ocorridas no banco de dados desde o último backup completo.
   - apenas do log (Transaction Log): uma das razões para se fazer backup do log de transações é evitar que o mesmo fique cheio. Se você o configurou para crescer automaticamente, não se preocupará com isso, a não ser que o disco onde está armazenado o log de transações esteja ficando cheio. Esta opção é desabilitada se **Truncate Log on Checkpoint** estiver ativa para o banco de dados em questão.
   - de algum arquivo ou grupo de arquivos (File and filegroup): esta opção também fica desabilitada se **Truncate log on checkpoint** estiver ativada para esse banco de dados. Um backup de grupo de arquivos (filegroup backup) copia apenas alguns dos arquivos físicos que formam o banco de dados. Usa-se esta opção quando não se dispõe de tempo para fazer backup de todo o banco de dados.
3. Clique em Add para para selecionar um destino (em fita ou em disco) para o backup.

4. Digite o nome do arquivo destino do backup, ou selecione um dispositivo de backup existente. No nosso caso, selecionaremos o dispositivo backup1, criado anteriormente. Clique em Ok.
5. Você pode optar por uma das ações a seguir (deixe a opção default - Append to media):
    - **Append to media**: com esta opção selecionada, este backup do banco de dados será adicionado a outros backups já existentes na fita, dispositivo ou arquivo selecionado.
    - **Overwrite existing media** fará com que este backup subsitutua, sobrescreva, o conteúdo da fita ou arquivo para onde ele está sendo gravado.
6. Clique em Ok para inciar o backup.

Ao clicar na guia Options, você pode optar por fazer uma verificação completa do backup após sua realização [Verify backup upon completion]. No caso de fazer backup para fita, você também pode definir uma data de expiração para o backup.
Para conferir o conteúdo de 'Backup1', selecione Backups, selecione-o na lista de dispositivos (do lado direito da jnela), dê um duplo clique no mesmo e clique no botão "View Contents".



Você verá qual o conteúdo atual, que inclui o banco de dados Exemplo.

Aí você vê os backups armazenados no arquivo ou dispositivo, com detalhes sobre o mesmo. Você vê aí, o nome do backup [Name], de qual servidor foi feito [Server], de qual banco de dados [Database], que tipo de backup [Type]. Se você rolar o conteúdo da janela horizontalmente, verá ainda a verá a data [Date] em que foi feito, o tamanho [Size], a data de expiração [Expiration] (disponível selecionando backup em fita), e alguma descrição [Description] que você tenha colocado. Clique em **Close** para sair dessa tela.

Acrescentando um backup

Você pode acrescentar um backup a um dispositivo. Vamos acrescentar o backup de exemplo2 (ou outro banco de dados qualquer que não seja o *exemplo*) a esse dispositivo. Para isso, faça o mesmo processo anterior (agora com o outro banco de dados), selecionando a opção **Append to media**, e clique em "Ok". Note que aqui você também pode ver o conteúdo de um arquivo ou dispositivo, clicando em **View Contents**.
Agora dê um duplo clique em "Backup1" na lista de backups, e no botão "View Contents". Você verá que lá dentro estão os dois bancos de dados.

Fazendo backup do log de transações

Um backup do banco de dados inteiro pode tomar muito tempo. Você pode fazer um backup do log de transações, que vai copiar apenas as modificações feitas.
No Enterprise Manager, selecione o banco de dados "Exemplo" , clique em **All Tasks**, **Backup Database**. Selecione a opção **Transaction Log**, escolha o dispositivo Backup1, e selecione **Append to media**. Clique em Ok.
Visualize a informação de 'Backup1' com o botão "View Contents". Note, na coluna "Backup Size", que o tamanho do log é menor do que o banco de dados.
Ao fazer esse tipo de backup, o conteúdo do log é limpo (exceto as alterações ainda pendentes no banco de dados). Isso signifca que você deve manter os backups de log para poder restaurar um banco de dados.

Limpando o log de transações

Quando o log de transações não é limpado freqüentemente, seu espaço pode se esgotar. Isso pode acontecer caso não sejam feitos backups freqüentes do log.
Se o espaço do log se esgotar, não é possível alterar dados no banco de dados. Para limpar o conteúdo do log (só as transações já confirmadas e escritas no log), use o comando: BACKUP LOG WITH TRUNCATE_ONLY ou, no Enterprise Manager, clique com o botão direito no banco de dados, **All Tasks**, **Truncate Log**. Aparece uma mensagem de confirmação, avisando que é altamente recomendável um backup do banco de dados completo antes de truncar o log de transações. Clique em Ok.

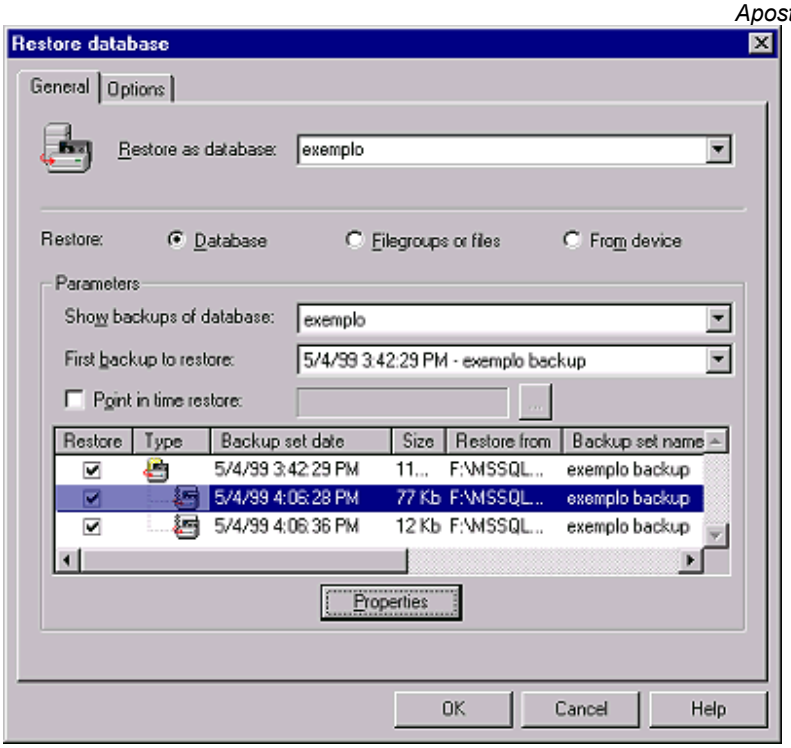
Restaurando um Backup

Você pode restaurar um backup através do Enterprise Manager.

Restaurando um banco de dados e o log pelo Enterprise Manager

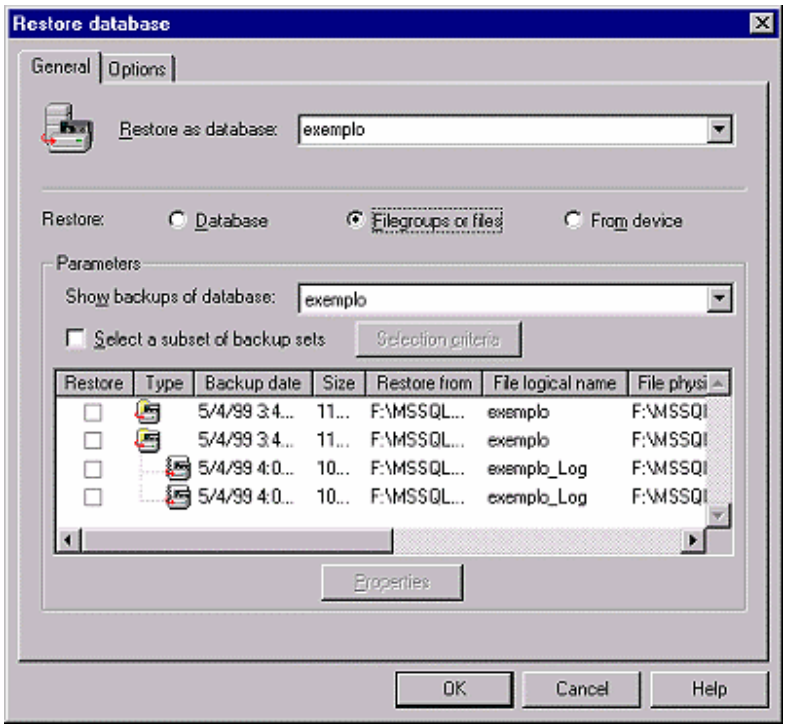
O Enterprise Manager te oferece uma caixa de diálogo que pode ser usada para restaurar um banco de dados. Para chegar a ela, selecione o banco de dados que você quer restauar, clique no mesmo com o botão direito, selecione All Tasks | Restore Database. A aparência da parte de baixo dessa caixa de diálogo depende da opção de restauração selecionada na metade superior, opção Restore.

- Caso você selecione Database, verá uma figura semelhante à número 1.
- Se selecionar Filegroups or Files, verás uma figura semelhante à número 2.
- Caso seleciones From Device, verás algo parecido com a figura número3.



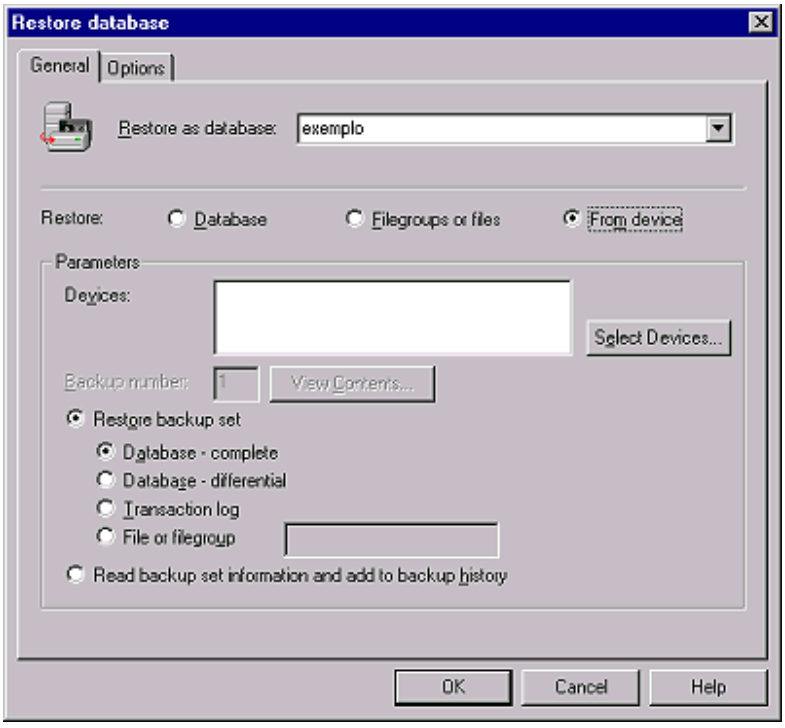
1

A primeira opção, "Restore Database", que mostra a figura acima, exibe uma lista dos backups a partir dos quais você pode pode pode fazer a restauração, baseado em tabelas de histórico armazenadas no SQL Server. Por padrão, o backup completo do banco de dados mais recente é mostrado primeiro, seguido pelos backups do log de transações e diferenciais realizados depois dele. Você pode usar a barra de rolagem para selecionar um backup completo feito há mais tempo. Note a sutil diferença entre os ícones de um backup completo do banco de dados e do backup do log de transações.
Nota: você pode saber que tipo de backup está representado pelo ícone, entre outros detalhes, clicando em Properties. Pode ser importante, pois um backup de grupo de arquivos tem o mesmo ícone que um backup completo do banco de dados.



2

A segunda opção, "Restore Filegroups or Files", também exibe uma lista de backups a partir dos quais você pode fazer a restauração com base nas tabelas de histórico. Junto com uma lista de backups anteriores de grupos de arquivos, você verá que backups de arquivos completos e de logs de transações. Eles são mostrados pois um grupo de arquivos também pode ser restaurado a partir de um backup completo do banco de dados e do log de transações.



3

A terceira opção, "Restore from Device", é utilizada quando o backup que você quer restaurar não está listado na tabela de histórico (os backups mostrados pelos ícones citados acima). Isso pode ocorrer se você estiver restaurando um banco de dados copiado de outro servidor SQL Server.

Nota: Só pelo fato de um backup estar listado, não significa que você seja capaz de realizar uma restauração através dele. As tabelas de histórico no SQL Server registram onde foi feito o backup do banco de dados, quando ele foi executado. Se desde então, os arquivos foram sobrescritos ou excluídos, você não será capaz de fazer uma restuaração a partir desse backup.

Quando você estiver fazendo a restauração, o primeiro arquivo selecionado deve ser um backup completo do banco de dados. A caixa de listagem "First backup to restore", lista os backups completos conhecidos. A caixa de diálogo mostrará os backups junto dos logs de transações e backups diferenciais subsequentes. O SQL Server seleciona todos ou alguns dos backups diferenciais e de log de transações a serem recuperados, em conjunto com o backup completo, automaticamente. A combinação selecionada pelo SQL Server fornece a imagem mais atualizada do banco de dados com o menor tempo de recuperação possíve;.

Se você não quiser restaurar também os backups do log de transações, você pode desselecioná-los clicando na caixa de verificação à esquerda de cada log. Se você desselecionar algum log, os logs subsequentes também não serão mais selecionados, automaticamente. Você não pode saltar nenhum log de transações no processo de restauração.

Se um backup diferencial do banco de dados (indicado pelo ícone)tiver sido realizado, ele também será selecionado, junto com o backup completo do banco de dados. Apenas o backup diferencial mais recente será selecionado.

Caso você queira restaurar o banco de dados para um ponto específico no tempo, você deve selecionar um log de transações, e então selecionar a caixa de verificação "Point in Time Restore" e informar a data e hora desejadas para a restauração na caixa de diálogo que aparece. Entào o banco de dados retornará ao estado em que se encontrava na data e hora selecionadas.

Restaurando backups de log

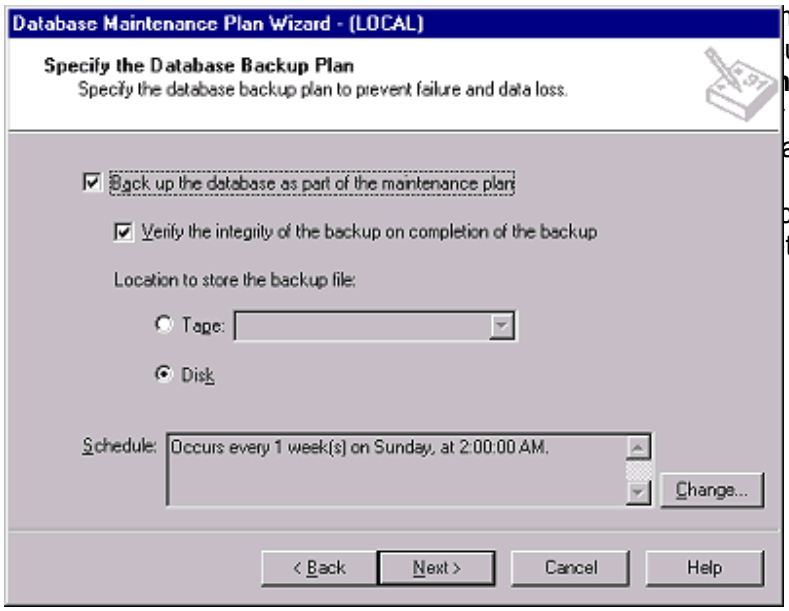
Um backup de log contém apenas as modificações feitas desde o último backup completo. Por isso, para restaurar um backup do log (indicado pelo ícone), você deve restaurar o backup do banco de dados mais recente anterior a ele.
Por exemplo, suponhamos que você faz um backup completo toda segunda-feira, e um backup do log nos outros dias. Se você quer restaurar um backup, deve recuperar o backup completo, e depois restaurar cada um dos backups de log, na ordem em que foram feitos.

Agendando Backups Automáticos

Backups devem ser executados frequentemente e regularmente. A melhor maneira de fazer isso é criar um trabalho no SQL Server para que o SQL Server Agent execute-o regularmente. Como a realização de backups afeta o desempenho do SQL Server, geralmente é melhor realizá-los durante períodos de baixa atividade, por exemplo nas primeiras horas do dia.
Importante: Para que qualquer tarefa agendada seja executada, o serviço SQL ServerAgent deve estar iniciado. Veja em Service Manager, uma das maneiras de se iniciar os serviços do SQL Server. Pode ser interessante definir que esse serviço inicie automaticamente na inicialização do computador.
Há três meios de se criar um trabalho de backup:

- Utilizar o assistente de plano de manutenção de banco de dados [Database Mainteinance Plan Wizard]: é recomendado porque também agenda outras atividades de manutenção de banco de dados que devem ser realizadas regularmente.
- Agendar um backup através da interface de bakcup imediato: a opção mais fácil. É como fazer um backup imediato (mostrado em Implementar um backup), exceto que se seleciona a opção **Schedule** (agendar) antes de se clicar em Ok.
- Escrever os comandos SQL você mesmo e criar a tarefa para executá-lo. Essa opção é a mais demorada para ser ajustada, mas é a que oferec maior flexibilidade.

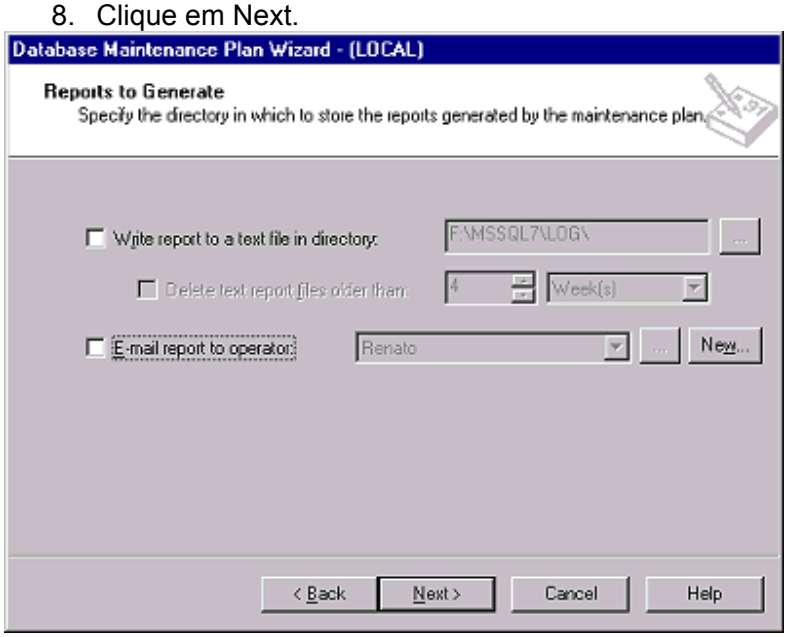
Agendando um backup completo de banco de dados ou de log de transações utilizando um assistente



m assistente, faça o seguinte:
ujo backup você quer agendar.
teinance Plan.
o plano de backup do banco de
abaixo:

ckup após seu término. Para isso,
tion of the backup".

4. Clique em Change para agendar quando o backup irá ocorrer. Note que o default é "Ocorre toda 1 semana no domingo, às 2:00 AM.". Ao clicar em Change, você verá a variedade de opções de agendamento disponíveis. Você pode definir data inicial e final (ou sem data final), ocorrer diariamente, semanalmente, mensalmente, ou a cada N semanas/meses/dias, a hora em que ocorrerá, o dia da semana, entre outras opções. Para o teste, clique em "Daily", deixe 1 dia e em "Occurs once", digite o horário desejado.
5. Escolha entre armazenar o backup em uma unidade de fita ou disco, e clique em Next.
6. Informe uma unidade de fita ou diretório de disco onde o backup deverá ser escrito. O processo de backup irá gerar automaticamente um nome de arquivo para o backup. Você pode optar por armazenar o backup de cada banco de dados em um diretório separado. Basta clicar em "Create a subdirectory ofr each database".
7. Se você quiser que o trabalho de backup exclua arquivos antigos do mesmo diretório para liberar espaço no disco rígido, opte por "remover arquivos mais antigos que" [Remove Files Older Than] e informe um número de semanas.
8. Clique em Next.



9. A próxima tela permite que você agende para que ocorra um backup também do log de transações ao ser realizada a tarefa. Para isso, selecione "Backup transaction log as part of the mainteinance plan". As opções disponíveis para backup do log de transações são as mesmas que para o backup do banco de dados (diretório, horários, etc...) Faça suas opções e vá clicando em Next até que apareça a tela de relatórios a gera [Reports to generate], mostrada abaixo:

   Nessa tela, você pode definir se será gerado um relatório da operação de backup, e informar o diretório onde o relatório será armazenado. Pode ainda definir um operador para receber por e-mail o relatório (para isso, você deve ter algum operador definido. Defina algum, expandindo, no Enterprise Manager, Management. Clique então com o botão direito em Operartors, e selecione New Operator. Informe os dados - nome, e-mail... - do operador, e clique em Ok.). Selecione da lista, o operador que irá receber um e-mail com o relatório. Clique em Next para continuar.

10. Aparece depois uma tela lhe informando onde será mantido o histórico da realização da tarefa, e você opde definir outro lugar para o mesmo. CLique em Next.
11. Por fim, lhe será mostrada uma tela exibindo as opções que você acabou de definir para o backup do banco de dados. Dê um nome à tarefa, e clique em Finish para que ela seja criada.

Também é possível agendar o backup completo de um banco de dados executadno os mesmos passos vistos na criação de um backup imediato, e selecionando Schedule. Clique no botão de reticências(**...**), para definir o agendamento do backup. É mais fácil e mais conveniente definir a tarefa do backup usando assistente de plano de manutenção de banco de dados, devido à variedade de opções disponíveis.

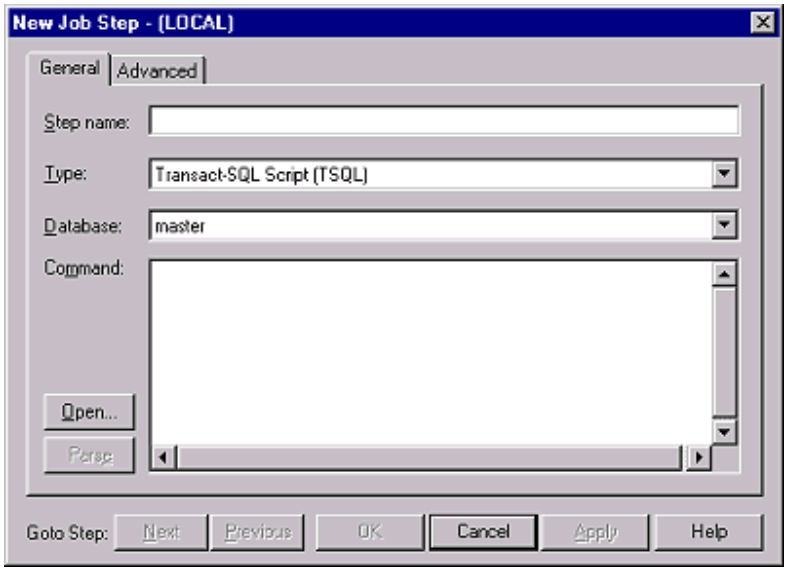
Como agendar um backup diferencial ou de grupo de arquivos

Para agendar um backup de grupo de arquivos ou diferencial, siga os passos para a criação de um backup imediato, e clique em Schedule, para definir que ele seja agendado. Clicando no botão de reticências (**...**) te permitirá escolher quando e com que frequênciao backup ocorrerá.

Criando uma tarefa para agendar um backup

É bastante recomendável utilizar assistentes para a criação de trabalhos de backup quando se está iniciando a trabalhar com o SQL Server, mas pode ser que você queira modificar tais trabalhos ou mesmo acrescentar um passo para realização de backup a um trabalho existente.Aqui então mostraremos um exemplo simples de como criar um trabalho a partir do zero, e como usar comandos SQL para realizar tal trabalho e backup.

1. No Enterprise Manager, expanda a pasta **Management**, expanda o **SQL Server Agent**, e clique com o obtão direito em **Jobs**. Selecione **New Job**.



2. Dê um nome para o trabalho e selecione a guia **Steps** (passos).
3. Escolha Clique no botão **New...** . Aparece a caixa de diálogo abaixo.

4. Na guia **General**, informe um nome para o passo, por exemplo "Backup do BD pubs".
5. Na lista **Type**, selecione "Transact-SQL Script". Isso lhe permitirá entrar com comandos SQL em **Command**.

6. O último passo é digitar o comando SQL a ser executado na caixa de texto Command. Para fazer backup completo do banco de dados *pubs* para um dispositivo de backup chamado *pubs_bk_dev*, o comando seria
   ```
   BACKUP DATABASE pubs RO pubs_bk_dev
   ```
   Para realizar um backup diferencial em *pubs*, o comando seria
   ```
   BACKUP DATABASE pubs TO pubs_bk_dev WITH DIFFERENTIAL
   ```
   Um backup de grupo de arquivos, supondo o arquivo authors no gruop de arquivos titles, do banco de dados pubs, seria feito com o seguinte comando
   ```
   BACKUP DATABASE pubs FILE = 'authors', FILEGROUP = 'titles' TO pubs_bk_dev
   ```

Você pode ver o histórico dos trabalhos de backup, selecionando o trabalho em questão no Enterprise Manager, e clicando no mesmo com o botão direito, selecionando **View Job History...**

14 - Outros Recursos

Configuração do SQL Server

Entendendo as Tabelas de Sistema

Importação e Exportação de Dados

Publicando dados na Internet

Objetivos:
- Entender as configurações do SQL e tabelas de sistemas;
- Fazer importação e exportação de dados usando o programa BCP;
- Aprender a publicar dados na Internet.

Configuração do SQL Server

O SQL Server tem várias opções de configuração, que geralmente podem ser deixadas com os valores default mas que, em alguns casos, podem ser alteradas para melhorar o desempenho ou a compatibilidade com os padrões ANSI.
Para alterar essas opções, use o Enterprise Manager. Selecione o servidor a ser configurado, clique no mesmo com o botão direito, e em **Properties**. Na primeira página, "**General**" estão informações gerais sobre o SQL Server, como o sistema em que ele está rodando, memória, processador, diretório raiz e as opções para auto-iniciar os serviços. Na página "**Security**" estão opções para selecionar o modo de autenticação (se integrada ao Windows NT, ou mista) e opções de auditoria relacionadas com a segurança integrada.
Na página "**Server Settings**" estão as opções de configuração do SQL Server. Elas são:

- default language for user: indica que língua será usada para mostrar as mensagens de sistema.
- Em **server behavior**, há três opções disponíveis:
  - "Allow modifications to be made directly to the system catalogs": se selecionada permite que mudanças sejam feitas à tabelas de sistema diretamente, através de comandos como INSERT, UPDATE, ou DELETE. Por padrão, isso não pode ocorrer. Mudanças às tabelas do sistema só são feitas através de procedimentos

armazenados de sistema. Cuidado!!! Permitir que tabelas de sistema sejam atualizadas diretamente pode fazer com que seu servidor não funcione adequadamente, ou nem sequer inicialize.]

- "Allow triggers to be fired which fire other triggers (nested triggers)": se selecionada, permite que se defina gatilhos que, em seu disparo, disparam outros gatilhos (gatilhos aninhados).
- "Use query governor to prevent queries exceeding specified cost": com essa opção selecionada, você pode definir um tempo limite, em segundos, para execução de consultas. Não é permitido executar consultas que ultrapassem tal tempo limite.

- A opção referente ao SQL Mail permite que você defina uma conta MAPI para o SQL Mail utilizar. Com o SQL Mail, mensagens podem ser enviadas por um gatilho ou porcedimento armazenado. Pode-se também processar consultas recebidas por e-mail e retornar os resultados criando um mail de resposta. Para maiores informações sobre como configurar o SQL Mail, consulte o capítulo "SQL Mail", em "Integrating SQL Server with other tools", dentro do livro "Administering SQL Server", no books online.
- Você também define aí como serão tratados os anos informados com dois dígitos.

Na página "**Connections**", você pode definir o número máximo de conexões simultâneas com o SQL Server [Maximum concurrent user connections]. O padrão é 0 (ilimitado). Também pode-se definir opções de conexão, em "Default connection options". Além disso, você pode permitir o estabelecimento de conexões remotas através de RPC [Allow other SQL serversto connect remotely to this server using RPC]; pode definir o tempo limite de uma consulta (o padrão é 0, que indica tempo ilimitado); por fim, pode forçar transações distribuídas.

Na página "**Database settings**" encontram-se algumas opções relativas a bancos de dados:

- a ção "default fill factor" permite que você determine qual o fator de preenchimento de páginas padrão quando da criação de um novo índice em dados existentes. Aí você tem duas opções: "Default (optimal)" que faz com que o SQL ajuste o fator de preenchimento automaticamente, e "Fixed" que te permite definir o fator de preenchimento (valores em porcentagem).
- A opção "Backup/restore" define quanto tempo o SQL Server deve esperar ao tentar ler uma fita de um dispositivo de fita: Indefinidamente [indefinitely], tentar uma vez e desistir [try once then quit], ou tentar durante N minutos [try for minutes].
- A opção "default backup media retention" especifica depois de quanto tempo o SQL Server vai reescrever em uma fita com backup. Ou seja, se você grava um backup hoje, e define esta opção para 7 dias, só depois de 7 dias é que o SQL Server vai escrever de novo nessa fita. Se você tentar escrever antes de decorrido esse prazo, receberá uma mensagem de erro.
- "Recovery interval (min)". Use esta opção para determinar o número máximo de minutos por banco de dados que o SQL Server precisa para recuperá-los. A cada vez que o SQL Server inicia, ele recupera cada banco de dados, desfazendo transações que não haviam sido finalizadas, e refazendo transações que haviam sido finalizadas, mas cujos dados não haviam sido escritos para o disco quando o SQL Server parou. Aqui você define um limite superior de tempo para recuperação de cada banco de dados. O padrão é 0, indicando configuração automática pelo SQL Server. Na prática, isso significa um tempo de recuperação de menos de um minuto e um checkpoint aproximadamente a cada minuto para bancos de dados ativos.
  O SQL Server estima quantas modificações de dados ele pode efetivar no intervalo, e faz um checkpoint no banco de dados quando o número de modificações de dados

feitas no banco de dados depois do último checkpoint alcança o número que o SQL Server estima que pode efetivar no intervalo.

Na página "**Memory**", você pode definir como quer que o SQL Server aloque memória para si. Você opde optar entre alocação dinâmica, sob demanda, de memória (a opção padrão), definindo aí a quantidade mínima e máxima de memória que o SQL Server pode alocar (por padrão, o mínimo é 0 e o máximo é a memória total do sistema); ou então optar por determinar uma quantidade fixa de memória a ser usada pelo SQL Server [Use a fixed memory size], que sempre estará alocada para o mesmo, e definir em quanto ficará fixada essa quantidade de memória.A opção "Reserve physical emory for SQL Server", se selecionada, indica que não será feito sequer *swap* nas páginas de memória do SQL Server; as mesmas estarão sempre disponíveis em RAM. Não use esta opção se você permitir que o SQL Server aloque memória dinamicamente. Por fim, a opção "minimum query memory" indica a quantidade mínima de memória a ser alocada para a execução de uma consulta. O aumento desse valor pode resultar em desempenho melhor na realização de certos tipos de consultas.

Na página "**Processors**" você encontra opções relativas à otimização do SQL Server em máquinas multiprocessadas (seção Parallelism). Na seção "Processor control" da página Processors, você tem as opções "Boost SQL Server priority on Windows NT", que fará com que os processos do SQL Server tenham prioridade maior sobre os outros processos em execução (cuidado ao usar essa opção pois outras tarefas realizadas no NT onde o SQL Server está rodando podem ficar prejudicadas). A opção "max worker threads" te permite definir o número máximo de tarefas (*threads*) que estarão disponíveis para processos do SQL Server (o padrão é 255, que funciona bem na maioria dos casos. Às vezes, a diminuição desse número, pode melhorar o desempenho). Se você marcar a opção "Use NT fibers" você pode ter um aumento no desempenho, já que o escalonamento de tarefas (*threads*) poderá ser feito no modo usuário (pelo uso das *fibers*), sem a necessidade de torca de contexto para o modo kernel que o uso apenas de threads implica. Procure por "Thread and task architecture" no Books online para entender melhor isso.

Nota: algumas dessas opções só têm efeito depois de parar e reiniciar o serviço MS SQL Server. Outras delas só permitem alteração se você definir que sejam mostradas opções avançadas. Para isso, no Query Analyzer, digite

```
sp_configure 'show advanced options', 1
```

depois pare e reinicie o serviço MS SQL Server. Aí você poderá alterar qualquer das opções citadas.

Entendendo as Tabelas de Sistema

As tabelas de sistema contêm informação atualizada automaticamente pelo SQL Server, que não pode ser alterada diretamente. No entanto, elas são úteis para fazer consultas à própria estrutura do banco de dados. Essas tabelas são todas documentadas no help do SQL Server (Transact-SQL Help).

Consultando sysobjects

A tabela *sysobjects*, por exemplo, contém informações sobre todos os objetos do banco de dados. As seguintes colunas são úteis:

name	*varchar(30)*	O nome do objeto.
id	*int*	Um número de identificação. Pode ser obtido também com a função OBJECT_ID(*nome*).

type	*char(2)*	Tipo do objeto: U: tabela de Usuário, S: tabela do Sistema, V: Visão, P: procedimento, R: regra, D: default, TR: trigger [gatilho], C: restrição Check, K: chave primária ou restrição UNIQUE, F: Foreign key (chave estrangeira)
crdate	*datetime*	A data/hora em que o objeto foi criado.

Por exemplo, para saber se uma tabela chamada Cliente já existe e excluí-la caso exista, pode ser usado o teste:

```
if exists (select * from sysobjects where type='U'
   and name='Cliente')
     drop table Cliente
```

Para ver todos os nomes de objetos do banco de dados, use:

```
select type, name, id
from sysobjects
order by type
```

As suas tabelas aparecem perto final da lista, pois são do tipo 'U'.

A tabela syscolumns

A tabela *syscolumns* é relacionada com *sysobjects* e guarda informação sobre cada coluna de cada tabela. Também guarda informações sobre parâmetros de procedimentos armazenados. Seus colunas mais importantes são:

id	*int*	Identificador da tabela ou do procedimento armazenado. Faz referência a *sysobjects.id.*
colid	*int*	Número seqüencial da coluna dentro da tabela (ou do parâmetro dentro do procedimento).
type	*int*	Identifica o tipo de dados básico, que faz referência a *systypes.type*.
length	*int*	Tamanho da coluna ou zero se não se aplica.
usertype	*int*	Tipo de dados definido pelo usuário. Faz referência a *systypes.usertype*.
name	*varchar(30)*	Nome da coluna.

Por exemplo, para listar todas as tabelas de usuário e suas colunas, use:

```
select so.name Tabela, sc.name Coluna
from sysobjects so inner join syscolumns sc on so.id = sc.id where so.type = 'U'
```

Podem aparecer tabelas chamadas *dtproperties*. Essa tabelas existem para armazenar diagramas do banco de dados.

Noções sobre diagramas

Diagramas servem para permitir que se visualize graficamente a estrutura do banco de dados sem se alterar os dados do mesmo. Através dos diagramas, você também pode testar novas estruturas para as tabelas, sem alterar o banco de dados. Além disso, pode-se criar novos índices, tabelas e relacionamentos através deles.

Para criar um diagrama, você clica com o botão direito no banco de dados que terá seu diagrama criado (no Enterprise Manager, claro), seleciona New | Database Diagram. Aí aparece o assistente de criação de diagramas, que lhe pergunta quais tabelas você quer incluir no diagrama. Informe-as, e quando clicar em Next, aparecerá o diagrama mostrando o relacionamento entre as tabelas (caso exista). Quando você sai da janela de diagrama, você pode salvá-lo com um nome qualquer.

Para saber mais sobre diagramas, consulte "Database Diagrams", no capítulo "Creating and Maintaining databases" do books online

Importação e Exportação de Dados

O programa BCP é um utilitário de linha de comando (não gráfico) usado para importar ou exportar tabelas de/para o SQL Server. Ele reconhece formatos de texto, bem como formatos binários. Para exportar o conteúdo da tabela Cliente, por exemplo, para um arquivo chamado CLIENTE.TXT, use-o da seguinte forma, na linha de comando (console do Windows NT ou prompt do DOS no Windows 9x):

```
BCP Exemplo..Cliente OUT CLIENTE.TXT -c -S nome_do_servidor -U sa -P
```

No caso, 'Exemplo' é o nome do banco de dados, a opção OUT especifica que os dados serão exportados, CLIENTE.TXT é o nome do arquivo que será criado. A opção "-c" indica um arquivo no formato texto. Se fosse usado "-n", seria usado o formato "nativo", isto é, com dados binários. A opção "-S" especifica o nome do servidor (substitua *nome_do_servidor* pelo nome do seu servidor). Finalmente "-U" e "-P" especificam o nome de usuário e senha, respectivamente. Se a senha for vazia, "-P" deve ser a última opção na linha de comando. Note que as opções são *sensíveis ao caso* (maiúsculas e minúsculas são diferentes).
O arquivo CLIENTE.TXT pode ser alterado em qualquer editor, ou importado em outro banco de dados.
Se você executar BCP OUT sem usar "-c", ele irá questionar interativamente o tamanho de cada coluna a ser usada.
Para importar uma tabela, existem alguns detalhes. A importação é muito mais rápida (especialmente no caso de tabelas grandes) se o banco de dados permitir operação *sem log* [nonlogged], ou seja, inserir linhas sem atualizar o log de transações. Para ativar essa opção, no Enterprise Manager, clique com o botão direito no banco de dados e **Properties**. Depois clique na página "Options", marque "Select into/bulk copy" e clique Ok.
Para teste, crie uma nova tabela com a mesma estrutura da tabela Cliente chamada "TempCliente". Importe os dados executando:
Nota: Você pode criar uma cópia da tabela Cliente, clicando na mesma com o botão direito, depois em All Tasks | Generate SQL Scripts. Aí clique em Preview. Depois que terminar-se de gerar o script, clique em Copy, e o script SQL será posto na área de transferência. Basta depois colá-lo no Query Analyzer, mudando o nome da tabela, e executar o comando.Lembre-se de retirar as primeiras linhas (até o GO), que verificam se a tabela existe, e se existir, a excluem, posicione-se no banco de dados onde você quer a tabela criada.

```
BCP Exemplo..TempCliente IN CLIENTE.TXT -c -S nome_do_servidor -U sa -P
```

Se a tabela tivesse uma coluna IDENTITY, você deveria incluir também a opção "-E" para usar SET IDENTITY_INSERT *nome_tabela* ON durante a importação de dados.
Outras opções, especialmente úteis com tabelas grandes, são "-F *numero_linha*" e "-L *numero_linha*". Elas permitem especificar respectivamente, os números da primeira e última linha que serão importadas ou exportadas. A opção "-b *tamanho*" diz a quantidade de linhas que é enviada em cada "batch" (lote de atualizações). Pode ser aumentada em relação ao default.

Publicando dados na Internet

O SQL Server Web Assistant permite publicar dados do SQL Server em formato HTML, ou seja páginas Web. Essas páginas podem ser colocadas num servidor Web e visualizadas numa Intranet ou na Internet com um browser (programa que visualiza HTML). As páginas, a priori,

não são alteradas dinamicamente, mas com o uso do Assistente, você pode definir uma periodicidade de atualização da página.
Para executá-lo, no Enterprise Manager, selecione Tools | Wizards. Clique no sinal de mais (+) ao lado de Management, e selecione Web Assistant Wizard.
Aparece a tela de boas-vindas do Web Assistant Wizard. Clique em Next.
Nota: você também pode executar o Web Assistant, expandindo o servidor cujos dados se quer publicar, expandindo a pasta Management. Aí, clique com o botão direito em **Web Publishing**, e em **New Web Assistant Job**.

Select Database

Aqui você deve informar de qual bancos de dados serão selecionadas as tabelas e colunas. Escolha da lista o banco de dados desejado. Clique em Next para continuar.

Start a New Web Assistant Job

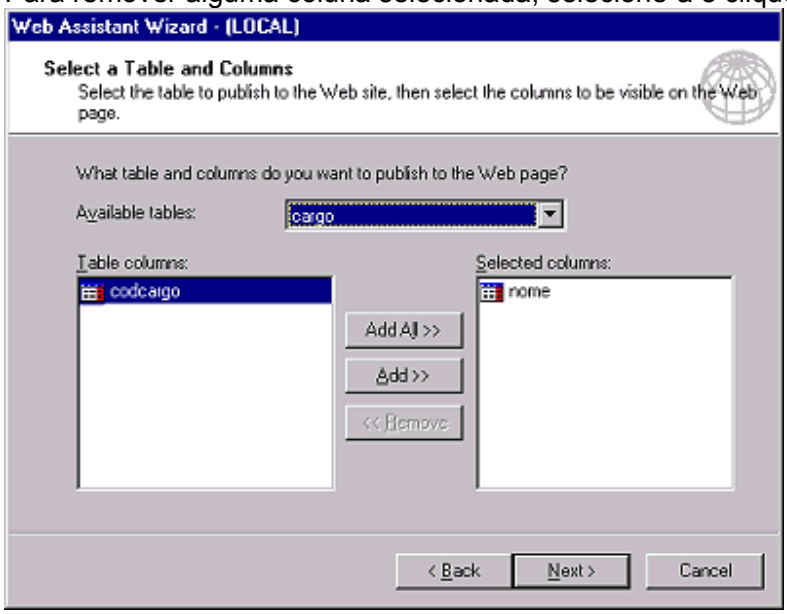
Informe um nome para o trabalho do Assistente Web, e informe de que forma virão os dados que você deseja publicar. As opções são as seguintes:

- "Data from the tables and columns I select": selecionando esta opção, lhe será permitido escolher quais colunas e tabelas fornecerão os dados para publicação.Caso você selecione esta opção siga para Seleção de tabelas e colunas.
- "Result set(s) of a stored procedure I select": mostrará os resultados da execução de um procedimento armazenado à sua escolha. Caso você selecione esta opção siga para Seleção de procedimentos armazenados.
- "Data from the Transact-SQL statement I specify": obterm os dados a serem mostrados através de uma sequência de comandos SQL que você informar. Caso você selecione esta opção siga para Informando comandos.

Feita sua seleção clique em Next para continuar. As próximas janelas a serem mostradas dependem do que você selecionou no passo acima.

Selecão de tabelas e colunas

Na tela mostrada abaixo, selecione as colunas que você deseja que sejam mostradas no seu resultado, selecionando-as do lado esquerdo da tela (embaixo de Table columns) e clicando em **Add**. Caso queira selecionar todas colunas da tabela em questão, clique em **Add All**. Note que as tabelas selecionadas aparecem do lado direito da tela, embaixo de "Selected columns". Para remover alguma coluna selecionada, selecione-a e clique em Remove.



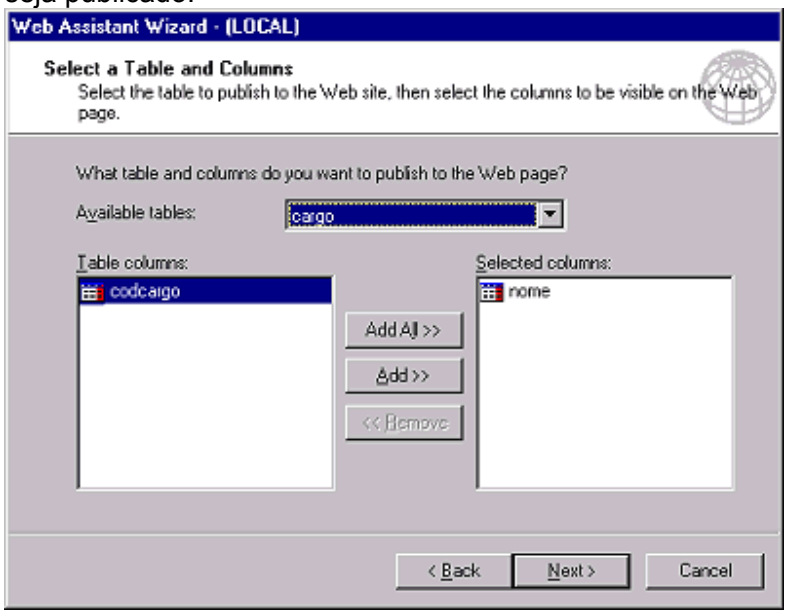
Caso queira exibir colunas de outras tabelas, selecione-as através da lista "Available tables". Clique em Next para continuar. Aparece a tela **Select rows**, onde você pode especificar o critério de seleção das linhas a serem mostradas. Você dispõe nessa tela de três opções, que são:

- All of the rows: mostrará todas as linhas da tabela que houver sido selecionada.
- "Only those rows that meet the following criteria": aqui você pode definir um critério para exibição das linhas, especificando uma coluna de uma tabela, um critério de comparação (=, <>, <, >, >=, <=), e um valor (número, string, o que for). Só serão exibidas as colunas que se enquadrarem no critério especificado.
- "Only those rows that qualify using the following Transact-SQL WHERE clause": só serão exibidas as linhas que atendam à declaração WHERE que você informar na caixa de texto logo abaixo.

Clique em Next e vá para Schedule the Web Assistant job.

Seleção de procedimentos armazenados.

Na tela mostrada abaixo, selecione o procedimento armazenado cujo resultado você quer que seja publicado.



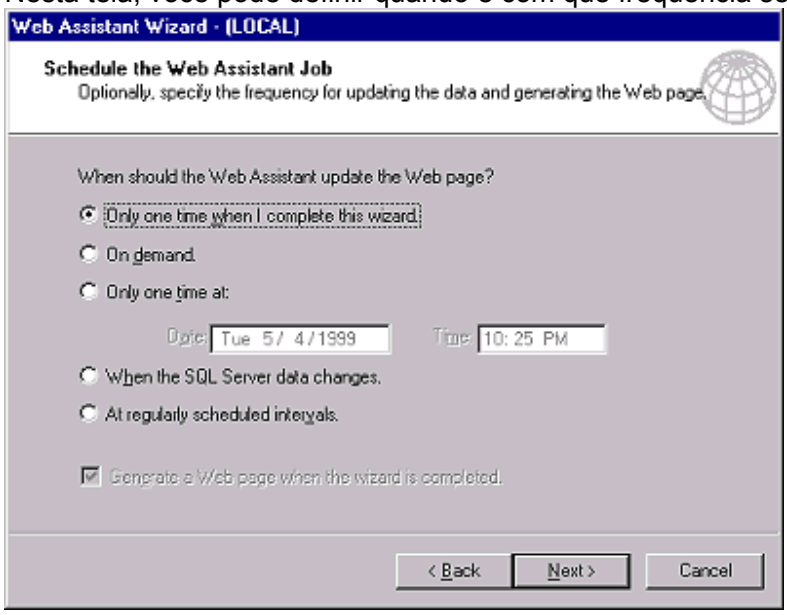
Clique em Next. Lhe será pedido para informar valores para todos os parâmetros necessários para o funcionamento do procedimento armzaenado. Informe-os, clique em Next, e vá para Schedule the Web Assistant job.

Informando comandos

Digite na caixa de texto a sequência de comandos SQL que você quer que seja executada de modo a mostrar os resultados a serem publicados. Clique em Next e vá para Schedule the Web Assistant job

Schedule the Web Assistant job

Nesta tela, você pode definir quando e com que frequência será gerada a página Web.



Com a primeira opção selecionada, a página será gerada apenas uma vez quando o assistente for concluído (no nosso caso, deixaremos essa primeira opção selecionada). Quando você seleciona qualquer outra opção, você pode definir se será ou não gerada uma página quando da conclusão do assistente. (opção "Generate a Web page when the wizard is completed", se marcada, gera uma página quando completa o assistente.). As outras opções são:

On demand	[Sob demanda]. O trabalho é executado mais tarde. Deve-se executar o Web Assistant de novo para executar a tarefa
Only one time at:	[Apenas uma vez em]. A página só é gerada uma única vez, no dia e hora especificados.
When the SQL Server data changes	[Quando os dados do SQL Server mudarem] Quando uma ou mais coluna de uma tabela forem alterados. Essa opção cria um gatilho na tabela para notificar automaticamente o Web Assistant das alterações.
At regularly scheduled intervals	[Em intervalos regularmente agendados]. Por exemplo, a cada 2 horas, a cada 2 dias etc. Clique em Next e, para esta opção, lhe será peritido definir a periodicidade de tais intervalos.

Clique em Next para continuar.

Publish the web page

Você deve informar o nome do arquivo. Para o teste, mantenha *C:\MSSQL7*\HTML\WebPage1.HTM. Note que o caminho para o arquiv pode ser um caminho local, um caminho de rede, ou até um caminho de um servidor FTP.Clique em Next.

Format the Web page

Agora, você pode obter a ajuda do assistente para formatar a página [Yes, help me format the web page] ou criar o arquvo a partir de um modelo [No, use template file from...") que é usado para definir a formatação da página. Caso você opte por usar a ajuda do assistente para formatar a página, selecione o conjunto de caracteres a ser usado. Para nosso teste, deixe o padrão [Universal Alphabet (UTF-8)]. Clique em Next.

Specify titles

Aqui você define o título da página [What do you want to title the Web page?], e o título da tabela HTML que conterá os dados [What do you want to title the HTML table that contains the data?]. Vamos informar o título da página: "Testando o Web Assistant" e o título para o resultado: "Relação de Clientes".
Você ainda pode definir que tamanho terá o título da tabela [What size should the HTML table title font be?] (H1, H2, H3..). Pode ainda escrever a data e hora de publicação da página [Apply a time and date stamp to the Web page].
Clique em Next.

Format a table

Aqui, você decide se quer os nomes das colunas sendo exibidas na tabela HTML. [Do you want column names displayed in the HTML table?]. Ainda é possível definir as características da fonte dos dados da tabela [What font characteristics do you want to apply to the table data?]. Por fim, pode optar por desenhar bordas ao redor da tabela [Draw border lines around the HTML table].
Clique em Next.

Add Hyperlinks to the Web Page

Agora, você pode incluir um ou mais links para outras URLs (endereços Internet), com várias opções:

- "No". Não incluirá nenhum link para URLs nas sua página.
- "Yes, add one hyperlink": especifique a URL que você quer "linkar" e um rótulo para esse link [Hyperlink label].
- "Yes, add a list of hyperlink URLs. Select them from a SQL Server table with the following SQL statement": você pode ainda informar as URLs dando um SELECT em colunas de tabelas dos seus bancos de dados. Digite os comandos SQL na caixa de texto.

Limit Rows
Pode-se limitar o número de linhas a serem exibidas na página, com algumas opções. As primeiras duas opções definem se serão exibidas todas [No, return all rows of data], ou as primeiras N linhas [Yes. Return the first rows of data.]
As outras duas opções lhe permitem definir se todos os dados serão postos em uma única página [No, put all data in one scrolling page], ou em diversas páginas ligadas por hiperlinks [Yes, link the successive pages togheter.]. No último caso, você deve informar quantas linhas serão exibidas por página.

Clique em Next.
Finalmente, você vê a tela final do assistente, lhe mostrando as opções deifnidas na criação dessa tarefa. Aí, você ainda tem a opção de escrever o código SQL gerador de tal tarefa para um arquivo [Write Transact-SQL to File...].
Clique em Finish.

Testando a página

Abra a página no Internet Explorer para visualizar o seu resultado.

Os links abaixo contém os scripts SQL para facilitar o acompanhamento da apostila. Abra-os e visualize no sew "browser". Para utilizá-los, selecione-os, copie e cole para o Query Analyzer.

Para voltar para esta página, use o botão BACK de seu browser, ou o menu de navegação à esquerda na tela.
Execute-os de preferência, em sequência.

Definindo novos tipos de dados

*** Adicionando**

```
sp_addtype 'TCEP', 'char(10)', 'null'
GO
sp_addtype 'TEstado', 'char(2)', 'null'
GO
sp_addtype 'TQuantidade', 'numeric(10,2)', 'not null'
GO
sp_addtype 'TTelefone', 'varchar (20)', 'null'
GO
sp_addtype 'TTipoOperacao', 'SmallInt', 'null'
go
sp_addtype 'TValorGrante', 'numeric(15,2)'
```

*** Removendo**

```
sp_droptype TTipoOperacao
GO
sp_droptype TValorGrante
```

Criando Tabelas

```
/****** Table CategoriaContato  ******/

CREATE TABLE CategoriaContato (
   CodCategoria  int NOT NULL,
   Nome          varchar (60) NULL
)
GO


/****** Table Contato   ******/

CREATE TABLE Contato (
   Tipo          char(1) NOT NULL CHECK (Tipo in ('P','E')),
   Codigo        int NOT NULL,
   CodigoSub     int NOT NULL,
   Nome          varchar (60) NULL
)
GO


/****** Table Empresa ******/

CREATE TABLE Empresa (
   CodEmpresa    int NOT NULL,
   Nome          varchar (60) NULL,
   RazaoSocial   varchar (60) NULL,


   -- campos adicionais
   DataCadastro  datetime NULL,
   Notas         text NULL
)
GO


/****** Table Produto ******/

CREATE TABLE Produto (
  CodProduto      int NOT NULL,
  Nome            varchar(60) NULL,
  Descricao       varchar(60) NULL,
 QuantDisponivel TQuantidade NOT NULL CHECK (QuantDisponivel >= 0)
DEFAULT 0,
  QuantMinima     TQuantidade NULL,
  Localizacao     varchar (50) NULL,
  Preco           money NOT NULL CHECK (Preco > 0)
)
GO

/****** Table MovimentacaoProduto ******/
```

```
CREATE TABLE MovimentacaoProduto (
   CodMovProduto  int NOT NULL,

   -- chave estrangeira da tabela Contato
   TipoContato    char(1) NULL,
   CodContato     int NULL,
   CodSubContato  int NULL,

   -- chave estrangeira da tabela Produto
   CodProduto     int NOT NULL,

   Quantidade     TQuantidade NOT NULL,
   DataMov        datetime NOT NULL DEFAULT (getdate()),
   -- E = Entrada, S = Saída
   TipoMov        char(1) NOT NULL CHECK (TipoMov in ('E','S'))
)
GO

/****** Table Pessoa ******/

CREATE TABLE Pessoa (
  CodPessoa    int NOT NULL,

  Nome         varchar (50) NULL,
  Sexo         char(1) NOT NULL,
  Fone         TTelefone NULL,
  Fax          TTelefone NULL
)
GO


/****** Table Subdivisao ******/

CREATE TABLE Subdivisao (
  -- chave primária
  CodEmpresa    int NOT NULL,
  CodSubdivisao int NOT NULL,
  Nome          varchar (50) NULL,
  Fone          TTelefone NULL,
  Fax           TTelefone NULL,

  -- colunas de endereço
  Rua           varchar (50) NULL,
  Bairro        varchar (25) NULL,
  Cidade        varchar (40) NULL,
  Estado        TEstado NULL,
  CEP           TCEP NULL,

  CGC           varchar (18) NULL,

  -- colunas adicionais
  DataCadastro  datetime NULL,
  Notas         text NULL
)
GO


/****** Table RelEmpresaCategoria ******/

CREATE TABLE RelEmpresaCategoria (
```

```
  CodEmpresa   int NOT NULL,
  CodCategoria int NOT NULL
)
GO

/****** Table RelPessoaCategoria ******/

CREATE TABLE RelPessoaCategoria (
  CodPessoa    int NOT NULL,
  CodCategoria int NOT NULL
)
GO
```

```
/****** Table RelSubdivisaoPessoa ******/

CREATE TABLE RelSubdivisaoPessoa (
  CodEmpresa    int NOT NULL,
  CodSubdivisao int NOT NULL,
  CodPessoa     int NOT NULL,

  Cargo         varchar (30) NULL
)
GO

/****** Table MovAcumulado ******/

CREATE TABLE MovAcumulado (
  CodProduto    int NOT NULL,
  TotalVendas     TQuantidade,
  TotalCompras  TQuantidade
)
GO

/**** Table Temporaria *****/
CREATE TABLE Temporaria (
  Codigo          int NOT NULL,
  Nome         varchar (50) NULL
)

/**** Table Temporaria1 *****/
CREATE TABLE Temporaria (
  Codigo          int NOT NULL,
  Nome         varchar (50) NULL
)
```

Altera, renomeia e exclui tabelas.

Alter Table

*** Campo**

```
ALTER TABLE Pessoa
 ADD  Rua    varchar(60) null,
      Cidade varchar(30) null,
      Bairro varchar(30) null,
      CEP    TCEP NULL,
      Estado TEstado NULL,
      CPF          varchar (14) NULL,
      DataCadastro datetime NULL DEFAULT (getdate()),
      Notas        text NULL
```

Renomear Tabela

```
sp_rename 'Temporaria1', TemporariaTeste
GO
sp_rename 'Temporaria.codigo', cod
```

Remover Tabela

```
drop table Temporaria, TemporariaTeste
```

Estarão criadas as tabelas, todas vazias. Deve-se executar o script DadosTodos, que colocará dados em todas elas.

Coloca dados em todas as tabelas (que já devem ter sido criadas).

Insere diversos dados nas tabelas já criadas

```
dump transaction Contatos
with truncate_only, no_log
GO

/**** Pessoa *****/

insert into Pessoa(CodPessoa,Nome,Sexo,Fone,Fax,Rua,Bairro,Cidade,Estado,CEP)
  values (1, 'Ab¡lio Frenkel', 'F', '222-2870', '031-295-4095', 'Av. Goi s, 345 sala
2', '', 'Goiƒnia', 'GO', ' ')
insert into Pessoa(CodPessoa,Nome,Sexo,Fone,Fax,Rua,Bairro,Cidade,Estado,CEP)
  values (2, 'AdÆo Dias', 'F', '261-8263', '222-4280', 'Rua C- 146, 661', '',
'Goiƒnia', 'GO', '')
insert into Pessoa(CodPessoa,Nome,Sexo,Fone,Fax,Rua,Bairro,Cidade,Estado,CEP)
  values (3, 'AdÆo Pereira', 'F', '234-7755', 'n0o lembra', ' Rua 2, 151', 'Setor
Universit rio', 'Goiƒnia', 'GO', '')
insert into Pessoa(CodPessoa,Nome,Sexo,Fone,Fax,Rua,Bairro,Cidade,Estado,CEP)
  values (4, 'Adebaldo Nepomuceno', 'F', '243-6465', '291-3257', 'Av. 24 de Outubro,
45', '', 'Goiƒnia', 'GO', '74215-030')
insert into Pessoa(CodPessoa,Nome,Sexo,Fone,Fax,Rua,Bairro,Cidade,Estado,CEP)
  values (5, 'Adriano Jesus', 'M', '281-6779', '295-4217', 'Av. Perimetral Norte,
3231', ' ', 'Goiƒnia', 'GO', ' ')
```

/**** Produto ****/insert into
Produto(CodProduto,Nome,Descricao,QuantDisponivel,QuantMinima,Localizacao,Preco)
values(1,"Lâmpadas 150 V","",4.00,2.00,"Estante , Prateleira , Divisão ",15.0000)insert into
Produto(CodProduto,Nome,Descricao,QuantDisponivel,QuantMinima,Localizacao,Preco)
values(2,"Toner/Laser Jet","",1.00,0.00,"Armário 01, Prateleira 02, ",15.0000)

Consultando dados, funções matemáticas, de caracteres, data/hora, conversão de dados e condições de pesquisa.

Consultando Dados

Consultando versão do SQL Server

```
Select @@version
```

Mostrando valor de uma string

```
Select 'Teste'
```

Consultando todas as colunas

```
* Mostrar quais são as pessoas existente no contatos?
Select * from pessoa
```

Consultando em outro banco de dados

```
* Mostrar quais são os autores cadastrados no bando de dados Pubs?
select * from pubs..authors
```

Consultando outras colunas

```
* Mostrar o nome, telefone e fax das pessoas cadastradas no contatos?
Select nome, fone, fax from pessoa
```

Consultando outras colunas, mudando o cabeçalho das colunas retornadas

```
* Mostrar as empresas cadastradas no contatos?
select nome 'Nome Fantasia', razaosocial 'Razão Social',
DataCadastro Cadastro from empresa
```

Usando condições

```
* Quais são as pessoas , cujo, estado  igual a 'Go' e cidade igual a 'Goiânia' ?
select Nome, Fone
from pessoa
where cidade = 'Goiânia' and estado = 'Go'
```

Manipulando Expressões

```
* Mostrar os preços de cada produto , após um aumento de 10%?
select nome Produto, Preco Preço, (preco * 1.1) "Preço com 10% de aumento"
from produto
```

Funções Matemáticas

```
* Mostrar os preços de cada produto , após um aumento de 10% arredondando o valor com
duas casas decimais?
select nome Produto, Preco Preço,
round((preco * 1.1), 2) "Preço arredondado"
from produto

select power(4,2), pi(), ascii('A')
```

Funções de caracteres

```
* Mostrar os produtos com as descrições?
select Nome + '-' +  descricao 'Produto/Descrição'
from produto

* Mostrar  os produtos  , onde os nomes podem aparecer com 20 caracteres e o preço com
6 casas decimais antes da vírgula e 2 casas depois da vírgula?
Select  substring(Nome, 1, 20) Produto , str(preco, 6, 2) Preço
from produto


* Repetir a letra a  10 vezes.
select replicate('a', 10)
```

Funções de Data/hora

```
select  datacadastro 'Data Cadastro',
dateAdd(mm, 1,datacadastro) 'Adicionando 1 Mês' ,
dateDiff(dd, datacadastro, getdate())
'Subtraindo Datas',
datepart(yy, datacadastro) Ano,
datepart(dw, datacadastro) Semana
from empresa
```

Conversão de Dados

```
select convert(char(10), nome) 'Nome Subdivisão',
       convert(char, datacadastro, 103)
       'Data Formato Brasileiro'
from subdivisao
```

Condições de Pesquisa

```
* Quais são os produtos, cuja quantidade disponível em estoque é menor que a
quantidade mínima permitida?

Select nome , quantdisponivel, quantMinima
from produto
Where quantDisponivel < quantMinima


* Quais as pessoas que não foi informado o telefone?

Select nome from pessoa
where fone is null


* Quais as subdivisões que estão no estado de Goiás e Tocantins?

select nome, fone, estado
from subdivisao
where estado in ('Go', 'To')

* Quais as saídas realizadas no período de 01/06/98 a 04/06/98?

select codproduto, quantidade, datamov
from movimentacaoproduto
where dataMov between '06/01/98' and '06/04/98'


* Quais as pessoas que contém as letras 'cris' no meio (ou no início ou no fim)?

select nome from pessoa
where nome like '%cris%'


* Quais as cidades onde a empresa possuí clientes, idependente se seja pessoa ou
subdivisão?

select cidade from pessoa
union
select cidade from subdivisao
```

Inserindo Linhas

```
INSERT INTO Pessoa
```

```
VALUES (400, 'PESSOA UM', 'M', '222-2222', '', 'R. Teste', 'B. DOS LIMOEIROS',
'Goiânia', 'GO', '74090-123',
'2222222-21', '07/22/1998', 'Obs. Nenhuma')

INSERT INTO Pessoa(CodPessoa,Nome,Sexo)
VALUES (401,'Pessoa401','F')
```

* Insert com select

Para testar iremos criar a tabela copiaempresa com a mesma estrutura da tabela empresa

```
CREATE TABLE CopiaEmpresa (
   CodEmpresa    int NOT NULL,
   Nome          varchar (60) NULL,
   RazaoSocial   varchar (60) NULL,

   -- campos adicionais
   DataCadastro  datetime NULL DEFAULT (getdate()),
   Notas         text NULL
)
```

Copiar as empresas cujo nome seja maior que 'M'

```
insert into copiaempresa
select * from empresa
where nome > 'm'
```

Para incluir os dados numa tabela com a estrutura diferente faça:

Criar a tabela copiapessoa

```
CREATE TABLE CopiaPessoa (

  CodPessoa    int NOT NULL,

  Nome         varchar (50) NULL,
  Sexo         char(1) NOT NULL CHECK (Sexo in ('M','F')),
  Fone         TTelefone NULL
)
```

Copiar as pessoa cujo sexo = 'F'

```
insert into CopiaPessoa
select codpessoa, nome, sexo, fone from pessoa
where sexo = 'F'
```

Atualizando dados

*** Alterar o estado das pessoas para 'Go' , quando a cidade for igual a 'Goiânia'**

```
update Pessoa
set Estado = 'Go'
where Cidade = 'Goiânia'
```

Update com select

Atualizar o total de vendas e compras quando seu valor for null.

Mas não existe nenhum registro cadastrado na tabela movacumulado, portanto, iremos incluir todos os produtos nesta tabela para depois fazer as alterações.

```
drop table movacumulado

go
CREATE TABLE MovAcumulado (
         CodProduto int NOT NULL ,
         TotalVendas TQuantidade  NULL ,
         TotalCompras TQuantidade NULL
)

go
insert movacumulado(codproduto)
         select codproduto from produto
go

select * from movacumulado

UPDATE MovAcumulado
SET TotalVendas =
 (select sum(Quantidade)
  from MovimentacaoProduto mp
  where mp.CodProduto =
   MovAcumulado.CodProduto
  and mp.TipoMov = 'S'),
TotalCompras =
 (select sum(Quantidade)
  from MovimentacaoProduto mp
  where mp.CodProduto =
   MovAcumulado.CodProduto
  and mp.TipoMov = 'E')
where TotalVendas is null
      or TotalCompras is null

select * from movacumulado
```

Excluindo dados

*** Excluir todos os registros da tabela copiapessoa**

```
delete from copiapessoa
```

*** Excluir os registros da tabela copiaempresa , quando o código da empresa for igual a 3**

```
delete from copiaempresa where codempresa = 3
```

Exclusão usando subconsulta

Excluir as linhas da tabela copiaempresa que existe na tabela empresa

```
Delete from copiaempresa
where codempresa in (select codempresa from empresa)
```

Coloca dados na tabela de funcionários

```
drop table funcionario
go
create table funcionario
(
  codigo int identity,
  nome char(60),
  tipo char(1),
  salario money
)

go

insert into funcionario
values('func1', 'T', 300)
go

insert into funcionario
values('func2', 'P', 300)
go
insert into funcionario
values('func3', 'P', 1000)
go
insert into funcionario
values('func4', 'T', 400)
go
insert into funcionario
values('func5', 'T', 1200)
go
insert into funcionario
values('func6', 'P', 1300)
go
insert into funcionario
values('func7', 'P', 200)
go
insert into funcionario
```

```
values('func8', 'P', 600)
go
insert into funcionario
values('func9', 'P', 700)
go
insert into funcionario
values('func10', 'T', 800)
```

Dados de Resumo

*** Quantas pessoas estão cadastradas na tabela de pessoas?**

```
Select  count(*) from Pessoa
```

*** Quantas pessoas existe por categoria?**

```
select CodCategoria 'Código Categoria',
  Count(*) 'Quantidade Pessoa'
from RelPessoaCategoria
group by CodCategoria
```

*** Relação de produtos da empresa com sua quantidade total vendida e ordenado pela quantidade.**

```
select CodProduto, SUM(Quantidade)
from MovimentacaoProduto
where TipoMov = 'S'
and DataMov between '7/6/98' and '7/7/98'
group by CodProduto
order by SUM(Quantidade) DESC
```

Funções Agregadas

```
select max(quantidade) 'Maior Quantidade',
       min(quantidade) 'Menor Quantidade',
       count(quantidade) 'Quantidade  Total',
       avg(quantidade) 'Média',
       sum(quantidade) 'Total'
 from movimentacaoProduto
```

*** Qual a última venda realizada para cada código do produto?**

```
Select codproduto, max(datamov)
from movimentacaoproduto
where tipomov = 'S'
Group by codproduto
```

*** Qual a menor quantidade vendida por cada código do produto?**

```
Select codproduto, min(quantidade) from movimentacaoproduto
Where tipomovimentacao = 'S'
Group by codproduto
```

*** Qual a quantidade total de vendas realizadas para cada código do produto?**

```
Select codproduto, sum(quantidade)
from movimentacaoproduto
Where tipomov = 'S'
        Group by codproduto
```

Having

*** Qual a menor saída realizada para cada código do produto quando a quantidade vendida for maior que 51?**

```
Select codproduto, min(quantidade)
from movimentacaoproduto
Where tipomov =  'S'
Group by codproduto
        Having  sum(quantidade) > 51
```

Junções de Tabelas

*** Quais os produtos que foram vendidos na empresa?**

```
Select distinct p.nome
from movimentacaoproduto m , produto p
where m.codproduto = p.codproduto
and m.tipoMov = 'S'
```

*** Quais as pessoas que pertencem a categoria 'CL Clientes', mostrando as pessoas em ordem alfabética?**

```
Select p.nome
from Pessoa p, RelPessoaCategoria r, CategoriaContato c
where p.codpessoa =  r.codpessoa and c.codcategoria = r.codcategoria
and c.nome = 'CL Clientes' order by p.nome
```

Junção usando Inner Join

*** Quais as subdivisões existente para cada empresa?**

```
select e.Nome, s.Nome
from Empresa e inner join Subdivisao s on e.codEmpresa = s.codEmpresa
order by e.nome, s.nome
```

Junção cruzada ou irrestrita

```
select e.Nome, s.Nome
from Empresa e cross join Subdivisao s
order by e.nome, s.nome
```

Junção exterior

```
Select * from produto left join movimentacaoproduto
      On produto.codproduto = movimentacao.codproduto
```

Junção com mais de duas tabelas

*** Quais as empresas que pertencem a categoria 'CL Clientes', mostrando as empresas em ordem alfabética?**

```
select  e.nome
from RelEmpresaCategoria r
inner join  Empresa e  on r.codempresa = e.codempresa
inner join CategoriaContato c on r.codcategoria =    c.codcategoria
where c.nome = 'CL Clientes'
order by e.nome
```

*** Relação das empresas e subdivisões com os nomes dos funcionários?**

```
select  e.nome 'Empresa', p.nome 'Funcionários'
from RelSubdivisaoPessoa r
inner join Pessoa P on r.codpessoa = p.codpessoa
inner join subdivisao s on (r.codsubdivisao = s.codsubdivisao and r.codempresa =
s.codempresa)
inner join Empresa E on S.codEmpresa = E.codEmpresa
order by E.nome , P.nome
```

*** Listagem das empresas com nome, fax, bairro, cidade ,estado e que pertencem a categoria igual a 2?**

```
select s.Nome,Fax,Bairro,Cidade,Estado
from Subdivisao s
                 inner join Empresa e
                 on s.CodEmpresa = e.CodEmpresa
                 inner join RelEmpresaCategoria r
                 on r.CodEmpresa = e.CodEmpresa
where r.CodCategoria = 2
```

```
order by Estado, Cidade, Bairro, s.Nome
```

Sub-Consultas

*** Porcentagem da quantidade de um produto em relação ao total de quantidades dos produtos, que foram comprados pelo empresa?**

select CodProduto, 100.0 * sum(Quantidade) / (select sum(Quantidade) from MovimentacaoProduto where TipoMov = 'E') as Porcentagem from MovimentacaoProduto where TipoMov = 'E' group by CodProduto
Select Nome, (select sum(Quantidade) from MovimentacaoProduto m where TipoMov = 'S' and m.CodProduto = p.CodProduto) as TotalVendas, (select sum(Quantidade) from MovimentacaoProduto m where TipoMov = 'E' and m.CodProduto = p.CodProduto) as TotalCompras from Produto p order by nome

Para as próximas consultas iremos criar a tabela de funcionário, para isso executar os seguintes comandos:

create table funcionario (codigo int identity, nome char(60), tipo char(1), salario money)

Executar o script Coloca dados na tabela de funcionários para acrescentar dados na tabela de funcionários. A coluna tipo se for P indica professor e se for T indica técnico.

Teste de existência

*** Mostrar os professores somente se tiver técnicos com o salário igual a 1200.**

```
Select nome , salario
From funcionario
Where funcionario.tipo = 'P' and
Exists (select * from funcionario where tipo = 'T' and salario = 1200)
```

*** Quais as empresas que não tem pessoas cadastradas nela?**

```
Select CodEmpresa,CodSubdivisao,Nome
from Subdivisao s
where NOT EXISTS
  (select *
   from RelSubdivisaoPessoa rsp
   where rsp.CodEmpresa = s.CodEmpresa
   and rsp.CodSubdivisao = s.Codsubdivisao)
```

*** Qual a data da última venda de cada produto com suas respectivas quantidades?**

```
select codproduto, max(datamov),
 (select quantidade from
  movimentacaoproduto p
  where p.codproduto = mp.codproduto
  and p.datamov = max(mp.datamov))
from movimentacaoproduto mp
where tipomov = 'S'
group by codproduto
```

*** Relação das empresas e subdivisões com a quantidade de funcionários pertencente a cada empresa?**

```
select  e.nome 'Empresa', count(*)
from RelSubdivisaoPessoa r
inner join Pessoa P on r.codpessoa = p.codpessoa
inner join subdivisao s on (r.codsubdivisao = s.codsubdivisao and        r.codempresa
= s.codempresa)
inner join Empresa E on S.codEmpresa = E.codEmpresa
group by e.nome
```

Índices

(Índices clustered geralmente é utilizado quando não é feito muita inclusão de dados na tabela.)

```
CREATE UNIQUE CLUSTERED INDEX indNome ON  CategoriaContato(nome)

CREATE CLUSTERED INDEX indNome ON Empresa(nome)

CREATE INDEX indNome ON Pessoa(nome)
```

Identity

```
Create table Departamento
(
                  Codigo int not null identity,
                  Nome varchar(60)
)
```

Default

```
create default
    DataAtual as getdate()
go

sp_bindefault DataAtual, 'empresa.datacadastro'
exec sp_bindefault DataAtual, 'subdivisao.datacadastro'

Alter table  Subdivisao
  add default 'GO' for  Estado

Alter table  Pessoa
  add default 'GO' for  Estado
```

Regras

```
create rule RegraSexo as @sexo in ('F', 'M')
go
sp_bindrule RegraSexo, 'pessoa.sexo'

ALTER TABLE MovimentacaoProduto
add check (@TipoMov in ('E', 'S'))
```

Restrições

```
/****** Object:  Table CategoriaContato ******/
ALTER TABLE CategoriaContato
        ADD PRIMARY KEY NONCLUSTERED (CodCategoria)

/****** Object:  Table Contato ******/
ALTER TABLE Contato
        ADD CONSTRAINT PkContato PRIMARY KEY CLUSTERED (Tipo, Codigo, CodigoSub)

/****** Object:  Table Empresa ******/
ALTER TABLE Empresa
        ADD CONSTRAINT PkEmpresa PRIMARY KEY NONCLUSTERED (CodEmpresa)

/****** Object:  Table Produto  ******/
ALTER TABLE Produto
        ADD CONSTRAINT PkProduto PRIMARY KEY NONCLUSTERED (CodProduto)


/****** Object:  Table MovimentacaoProduto   ******/
ALTER TABLE MovimentacaoProduto
        ADD PRIMARY KEY NONCLUSTERED (CodMovProduto),
            FOREIGN KEY (CodProduto) REFERENCES Produto(CodProduto),
            FOREIGN KEY (TipoContato, CodContato, CodSubContato)
                REFERENCES Contato(Tipo,Codigo,CodigoSub)

/****** Object:  Table Pessoa ******/
ALTER TABLE Pessoa
        ADD PRIMARY KEY NONCLUSTERED (CodPessoa)

/****** Object:  Table Subdivisao ******/
ALTER TABLE Subdivisao
        ADD PRIMARY KEY NONCLUSTERED (CodEmpresa, CodSubdivisao)

/****** Object:  Table RelEmpresaCategoria ******/
ALTER TABLE RelEmpresaCategoria
```

```
        ADD CONSTRAINT PkRelEmpresaCategoria PRIMARY KEY NONCLUSTERED
(CodEmpresa,CodCategoria),
            CONSTRAINT FkCodEmpresaCategoria FOREIGN KEY (CodCategoria) REFERENCES
                CategoriaContato(CodCategoria)

/****** Object:  Table RelPessoaCategoria ******/
ALTER TABLE RelPessoaCategoria
        ADD PRIMARY KEY (CodPessoa, CodCategoria),
            CONSTRAINT FkCodPessoaCategoria FOREIGN KEY (CodCategoria) REFERENCES
                CategoriaContato(CodCategoria)

/****** Object:  Table RelSubdivisaoPessoa   ******/
ALTER TABLE RelSubdivisaoPessoa
        ADD PRIMARY KEY (CodEmpresa,CodSubdivisao,CodPessoa),
            FOREIGN KEY (CodEmpresa,CodSubdivisao) REFERENCES Subdivisao,
            FOREIGN KEY (CodPessoa) REFERENCES Pessoa
```

*** Criar estas tabelas que utiliza as restrições de integridade com o comando create.**

```
CREATE TABLE copiaPessoa (
        CodPessoa int NOT NULL constraint PK_Pessoa primary key,
        Nome varchar (50) NULL constraint U_NomePessoa Unique,
        Fone TTelefone NULL ,
        Fax TTelefone NULL ,
        Sexo char (1) check (sexo in('F','M')),
        Rua varchar (60) NULL ,
        DataCadastro datetime default getdate() ,
        Cidade varchar (30) default 'Goiânia' ,
        Bairro varchar (30) NULL ,
        CEP TCEP NULL ,
        Estado TEstado NULL ,
        CPF varchar (14) NULL ,
        Notas text NULL )


CREATE TABLE CopiaRelPessoaCategoria (
                CodPessoa int NOT NULL constraint
FK_CopiaPessoa  foreign key references copiapessoa,
                CodCategoria int NOT NULL
                primary key (codpessoa, codcategoria)
)
```

Visões

```
CREATE VIEW EmpresaCategoria
 (CodEmpresa, Empresa, Categoria)
AS
select e.CodEmpresa, e.Nome, c.Nome
from Empresa e
  INNER JOIN RelEmpresaCategoria rec
  ON e.CodEmpresa = rec.CodEmpresa
  INNER JOIN CategoriaContato c
  ON rec.CodCategoria = c.CodCategoria
```

```
CREATE VIEW ProdutoRestrito
AS SELECT CodProduto, Nome, Localizacao
   FROM Produto

CREATE VIEW PessoasEmpresa as
select p.Nome 'Pessoa', e.nome 'Empresa', s.nome 'Subdivisao', rs.cargo
from Pessoa p
     inner join RelSubdivisaoPessoa rs
        on p.CodPessoa = rs.CodPessoa
     inner join Subdivisao s
       on (rs.CodEmpresa = s.CodEmpresa
           and rs.CodSubdivisao = s.CodSubdivisao)
     inner join Empresa e
       on (s.codempresa = e.codempresa)


CREATE VIEW PessoaView
AS SELECT Nome, Cidade, Estado
   FROM Pessoa
   WHERE Nome like 'A%'

CREATE VIEW Saidas
AS SELECT DataMov, CodProduto, Quantidade, TipoMov
FROM MovimentacaoProduto
WHERE TipoMov = 'S'
WITH CHECK OPTION
```

Procedimentos

```
CREATE PROCEDURE BuscaPessoa
  @nome varchar(50)
AS
  declare @pesquisa varchar(50)
  declare @contagem int

  select @pesquisa = '%' + @nome + '%'
  select @contagem = count(*)
   from Pessoa
   where Nome like @pesquisa

  declare @mensagem varchar(100)

  IF @contagem != 0
  begin
    select @mensagem =
      convert(varchar,@contagem) +
      ' pessoas encontradas'
    print @mensagem

    select CodPessoa, Nome
    from Pessoa
    where Nome LIKE @pesquisa
    order by Nome
  end
  ELSE
  begin
    select @mensagem =
      'Não foi encontrado "'+@nome+'"'
```

```
    print @mensagem
  end
GO

/**************/

create procedure BuscaPessoaCategoria
  @nome varchar(50),
  @codCategoria int = 0
AS
  if @codCategoria = 0
  begin
   select * from Pessoa
   where Nome like '%'+@nome+'%'

   return 1
  end
  else begin
   select * from Pessoa
   where Nome like '%'+@nome+'%'
   and CodPessoa in
   (select CodPessoa from RelPessoaCategoria
    where  CodCategoria = @codCategoria)

   return 2
  end
GO

/**************/

create procedure ConsultaVendasProduto
 @codProduto int,
 @dataIni    datetime,
 @dataFim    datetime
as
select Quantidade,DataMov,
 TipoContato,CodContato,CodSubContato
from MovimentacaoProduto
where CodProduto = @codProduto
and   TipoMov = 'S'
and   DataMov between @dataIni
              and     @dataFim
order by DataMov

/**************/

create procedure InicializarContato
as
delete from Contato

insert into Contato
  select 'P', CodPessoa, 0, Nome
  from Pessoa

insert into Contato
  select 'S', CodEmpresa, CodSubdivisao,
         Nome
  from Subdivisao
GO

/**************/
```

```
create procedure AtualizarVendasCompras
as
delete from MovAcumulado

insert into MovAcumulado(CodProduto,TotalVendas,TotalCompras)
select CodProduto,
  Preco*(select sum(Quantidade)
         from MovimentacaoProduto m
         where m.TipoMov = 'S'
         and m.CodProduto = p.CodProduto),
  Preco*(select sum(Quantidade)
         from MovimentacaoProduto m
         where m.TipoMov = 'E'
         and m.CodProduto = p.CodProduto)
from Produto p
GO

/**************/

CREATE PROCEDURE IncrementaProduto
@codProduto int, @QuantAdicionada TQuantidade
AS

  update produto
  set quantdisponivel = quantdisponivel +
      @quantAdicionada
  where
     codproduto = @codproduto
  if @@Error != 0
       return @@Error
  else
       return 0
GO

/**************/

CREATE PROCEDURE DecrementarProduto
@CodProduto int, @QuantRetirada TQuantidade
AS
 if (select quantdisponivel
       from produto
       where codproduto = @codproduto) <
      @QuantRetirada
 begin
    RaisError('Quantidade Insuficiente!',
          16, 1)
    return 1
 end
 update  produto
 set quantdisponivel = quantdisponivel -
     @quantretirada
 where codproduto = @codproduto

 if @@Error != 0
   return @@Error
 else
   return 0
GO
```

Usar o procedimento Documentador , porque ele utiliza cursor e é um exemplo de como utilizar tabelas de sistemas.

```
create procedure Documentador
@tabela varchar(50) = NULL
as
if @tabela IS NULL
 declare SOBJ cursor for               //declarando cursor com nome SOBJ
   select id,name from sysobjects
   where type = 'U'
   order by name
else
 declare SOBJ cursor for
   select id,name from sysobjects
   where name = @tabela
declare @id int, @nomeTabela varchar(50)
declare @msg varchar(100)

// Depois de criado tenho que abrir o cursor , após a abertura ele fica posicionado no
início.

open SOBJ

fetch next from SOBJ into @id, @nomeTabela // busca o próximo registro e guarda esses
valores nas variáveis id, nometabela, para percorrer a tabela tem que colocar num
laço.

while @@fetch_status != -1
begin
  select @msg = '***** Tabela: ' + @nomeTabela
  print @msg
  select sc.colid Num, sc.name Coluna,
  convert(varchar(20),
   case
     when st.name in ("char", "varchar", "binary")
          then st.name + "(" + convert(varchar,sc.length)+")"
     when st.name in ("numeric", "decimal")
          then st.name + "(" + convert(varchar,sc.prec) + "," +
                               convert(varchar,sc.scale) + ")"
     else st.name
   end) Tipo,
   case when sc.status & 0x08 = 0 then "NOT NULL " else "NULL " end +
   case when sc.status & 0x80 != 0 then "IDENTITY" else null end
    Opções,
   sc.length 'Tam.bytes', sc.cdefault, sc.domain
   from syscolumns sc LEFT JOIN systypes st
   ON sc.usertype = st.usertype
  where sc.id = @id

  fetch next from SOBJ into @id, @nomeTabela
end
close SOBJ
deallocate SOBJ // destroi o cursor do banco de dados
GO
```

Modelo de exemplo

Aqui você tem a descrição de uma solução de banco de dados para uma empresa fictícia. Com base na realidade da empresa, deduzem-se os requisitos de sistema. Faz-se então o modelo de dados para o sistema.

Requisitos do Sistema

Contatos

Uma empresa precisa manter os dados dos seus "contatos", ou seja, das várias pessoas e empresas que se relacionam com ela, sejam eles clientes, fornecedores, distribuidores etc.
Para cada empresa cadastrada, é preciso manter seu nome e razão social. Cada empresa tem uma ou mais subdivisões. Uma subdivisão representa uma filial ou departamento da empresa, que pode ter uma localização física separada. Uma empresa não pode ser cadastrada sem nenhuma subdivisão.
Cada subdivisão pode ter um endereço, que é dividido em rua/número, bairro, cidade, estado e CEP. Além disso, uma subdivisão tem um telefone, opcionalmente um fax e opcionalmente um CGC.
Para cada pessoa, é preciso manter seu nome e sexo, além dos dados de endereço como na subdivisão. Uma pessoa pode ter, opcionalmente, um telefone, um fax e o seu número de CPF.
Uma subdivisão pode ter pessoas que trabalham nela, sendo necessário saber o cargo da pessoa. Uma mesma pessoa pode estar empregada em diferentes subdivisões, até mesmo de diferentes empresas.
As pessoas e empresas são classificadas em categorias, para maior organização. Cada pessoa ou empresa pode pertencer a mais de uma categoria ou a nenhuma categoria.
Ao cadastrar qualquer empresa, subdivisão ou pessoa, é preciso guardar a sua data de cadastramento e um texto opcional contendo informações adicionais.
Freqüentemente, os usuários precisam emitir um relatório das empresas e pessoas de uma determinada categoria, em ordem alfabética.
A empresa envia mensalmente um informativo para todos os seus clientes (sejam pessoas físicas ou empresas). O informativo é emitido via fax, apenas para aquelas pessoas ou empresas que estão na categoria "Clientes" e possuem número de fax. A lista de clientes deve ser ordenada por estado, depois por cidade e depois por bairro. Um sério problema do sistema anterior é que durante a consulta para essa mala direta (em horário de trabalho), todo o processamento ficava mais lento.

Produtos

A empresa mantém um controle dos produtos que ela comercializa. Para cada produto são mantidos um nome informal, uma descrição mais detalhada, a localização no estoque, a quantidade disponível em estoque e a quantidade mínima em estoque.
Cada entrada ou saída de produto deve ser registrada. Para uma entrada, é necessário saber qual o contato (empresa ou subdivisão) que forneceu o produto. Igualmente, para uma saída, é preciso saber qual o contato para o qual o produto foi vendido. Tanto para entradas como saídas, é preciso registrar a quantidade do produto e a data em que foi feita essa movimentação.
O sistema anterior da empresa, que usava arquivos DBF, tinha um cadastro de produtos. É importante que esse cadastro de produtos seja reaproveitado. Os programadores conseguiram

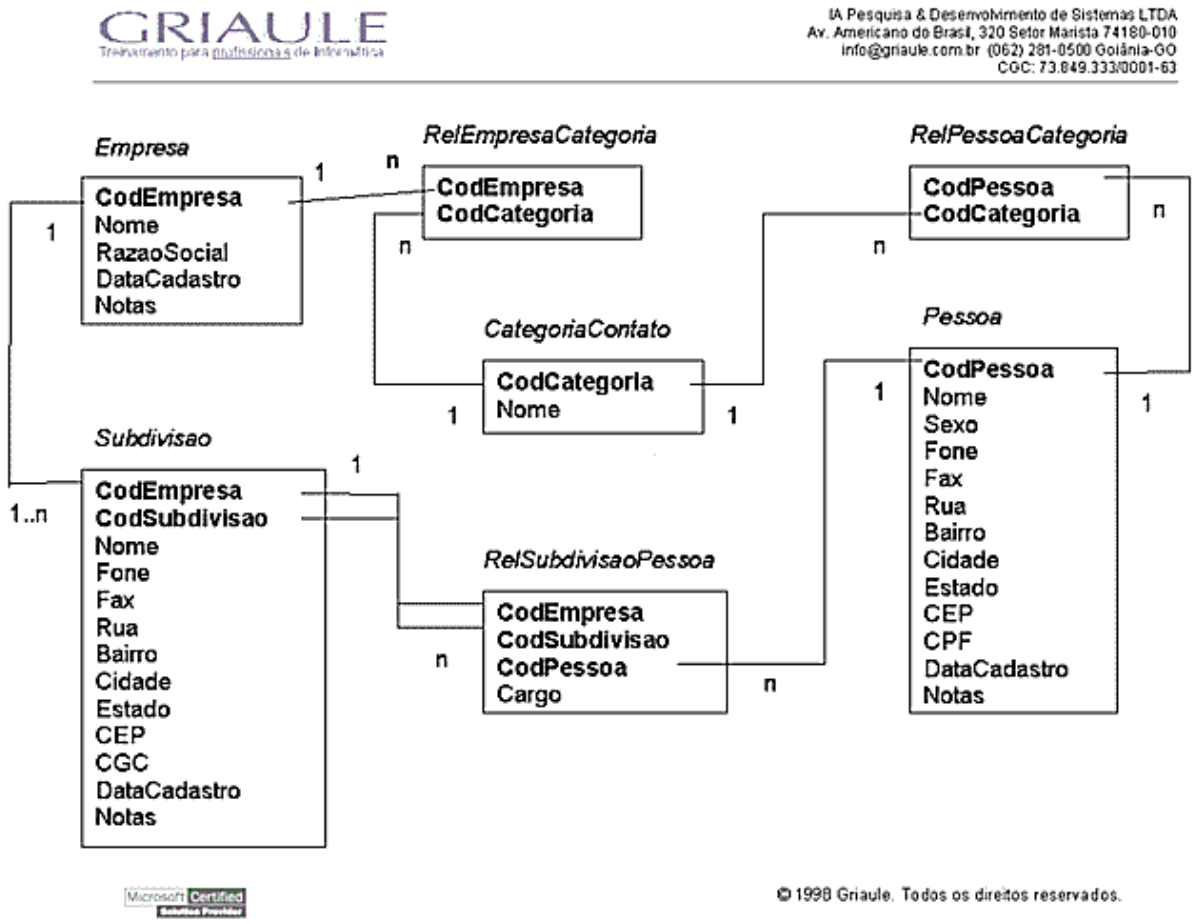
exportá-lo para um arquivo texto, separado por tabulações. No entanto, alguns campos são diferentes do novo sistema.
A gerência precisa dos seguintes relatórios:

- **Quais as vendas de cada produto em determinado período?**
- **Quais as vendas e compras de cada produto num determinado período? Por exemplo:**

Produto Total vendas Total compras
Primeiro Produto 120.050,30 30.456,00
Segundo Produto 3.457,10 2.000,87
... ...

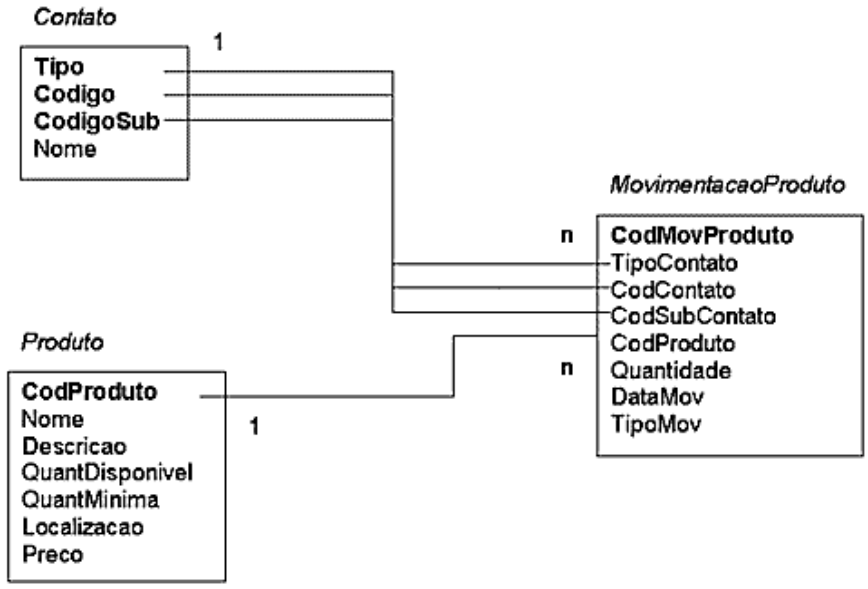
- Qual a última venda de cada produto?

Modelos de dados



GRIAULE
Treinamento para profissionais de Informática

IA Pesquisa & Desenvolvimento de Sistemas LTDA
Av. Americano do Brasil, 320 Setor Marista 74180-010
info@griaule.com.br (062) 281-0500 Goiânia-GO
CGC: 73.849.333/0001-63



UPGRADE do SQL Server 6.x para a versão 7.0

Antes de fazer o Upgrade do SQL Server 6.x para a versão 7.0, deve-se seguir estes passos:

- Executar os seguintes comandos DBCC em todos os bancos de dados existentes: CHECKDB, NEWALLOC, CHECKCATALOG.
- Confirmar que o parâmetro de configuração do SQL Server, bancos de dados abertos, é maior ou igual ao número de bancos de dados no servidor (inclusive master, tempdb, model...)
- Fazer cópias de segurança de todos os bancos de dados, inclusive o master. Se possível, usar a ferramenta de backup do Windows NT, pra fazer backup de todas as pastas do SQL Server.
- Fazer backup do registro do NT.
- Desligar bancos de dados que sejam somente leitura (read-only). Qualquer banco de dados que tenha a proprieade read-only em TRUE, passá-la para FALSE. Com o comando CHKUPG, você descobre quais os bancos de dados em modo de somente leitura.
- Fechar qualquer aplicação de SQLServer, e assegurar-se que ninguém está usando o SQLServer
- Fazer o upgrade!

Nota: Para atualizar de versões do SQl Server anteriores à 6.x, primeiro deve-se fazer o upgrade para 6.x, para só então atualizar para a versão 7.0.
Existem diversos cenários possíveis para o upgrade. O primeiro cenário é quando se está fazendo a atualização de uma instalação existente de SQL Server 6.x para 7.0 na mesma máquina. Aí, o SQL Server 7.0 vai ser instalado na máquina em conjunto com o 6.x. As duas versões não podem rodar simultanamente, mas podemos facilmente alternar entre os dois. Aí, usa-se o "SQL Server 7.0 Version Upgrade Wizard" para exportar os bancos de dados da versao 6.x para o disco, fita ou drives de rede e então importa-se tais bancos de dados para o SQL Server 7.0.
O outro cenário é quando você dispõe de dois computadores. Como não se está instalando o SQL Server 7.0 na mesma máquina que está instalado o SQL Server 6.x, tudo se torna mais fácil. Pode-se migrar todos os seus bancos de dados de uma vez ou alguns de cada vez. Você e seus usuários vão poder acessar o servidor 7.0 e o 6.x depois do processo de atualização porque os mesmos estão rm computadores diferentes (logo, podem rodar simultaneamente, se desejado).

Informações gerais sobre o Assistente de Atualização de Versão (Version Upgrade Wizard)

Se você tem dois computadores, como dito acima, você pode acessar os bancos de dados da versão 6.5 e da 7.0 depois que o processo de instalação tiver terminado. Quando se fizer a atualização em uma única máquina, o estado do seu SQL Server 6.5, depois que o Assistente de Atualização de Versão tiver terminado, depende do método escolhido para transferir os bancos de dados. Se você tiver espaço em disco suficiente no seu servidor para instalar o SQL Server 7.0 sem remover os dispositivos de dados (devices) do SQL Server 6.x, você pode usar o método Direct Pipeline para transferir os dados. A abordagem Direct Pipeline é a melhor escolha a ser feita quando se faz uma atualização. Com esse método, ocorre a transferência em memória dos dados e objetos do SQL Server 6.x para o SQL Server 7.0, deixando o SQL Server 6.x intacto. Com esse método, também se verifica a melhor performance no processo de ataulização. Mas, se você não tiver espaço em disco para fazer a atualização sem remover

seus bancos de dados do SQL Server 6.x primeiro, você terá que exportar os dados e objetos do SQL Server 6.x para uma fita ou drive de rede. A opção do drive de fita é a melhor, mas se não se dispuser de tal recurso, use a opção da unidade de rede.

Alternando entre o SQL Server 6.5 e o SQL Server 7.0

O SQL Server 6.5 e o SQL Server 7.0 não podem rodar simultanteamente na mesma máquina. Quando você instala o SQL Server 7.0 em uma máquina que já tem o SQL Server 6.5, é criado um item no menu Iniciar, Programas, Microsoft SQL Server - Switch. Se você estiver com o SQL Server 7.0 ativo, há um ícone Microsoft SQL Server 6.5; caso você esteja com o SQL Server 6.5 ativo, o ícone é Microsoft SQL Server 7.0. Ao clicar nesse ícone, aparece uma caixa de diálogo avisando que o SQL Server está restaurando as informações do SQL Server 6.5 (ou 7.0). Então você alterna entre um e outro SQL Server com facilidade. Esse switch cuida de parar e iniciar os serviços necessários para que cada uma das versões do SQL Server possa rodar.

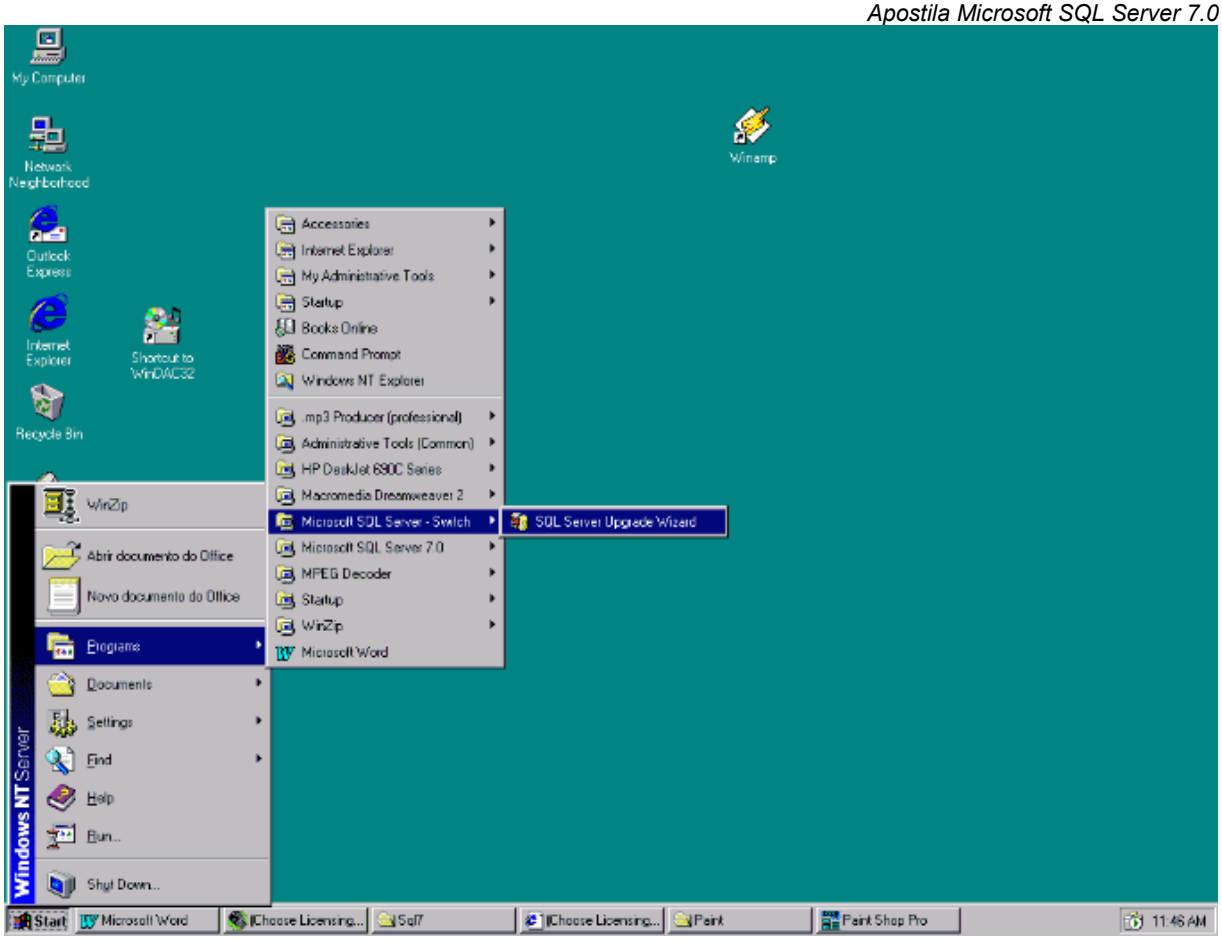
Asssistente de Atualização de Versão

Mesmo que não se tenha o SQL Server 6.5 instalado, se durante a instalação for selecionada a opção Upgrade Tools, vai ser instalado o Assistente de atualização de versão.
O Upgrade Wizard, quando se usa o método direct Pipeline, oferece alguma proteção contra falhas (ou seja, se deu problema, dá pra desfazer). Mas mesmo assim, é recomendável seguir os passos descritos no início do capítulo, e para enfatizar, não deixe de:

- Realizar backups dos bancos de dados;
- Fazer backups dos arquivos do SQL Server, dispositivos (devices) e diretórios;
- Fazer backup do o registro do NT;
- Ter espaço suficiente no disco pra atualização.
- Ajustar o TempDB para 25MB ou mais.

Depois que você tiver certeza que fez tudo isso, podemos começar!

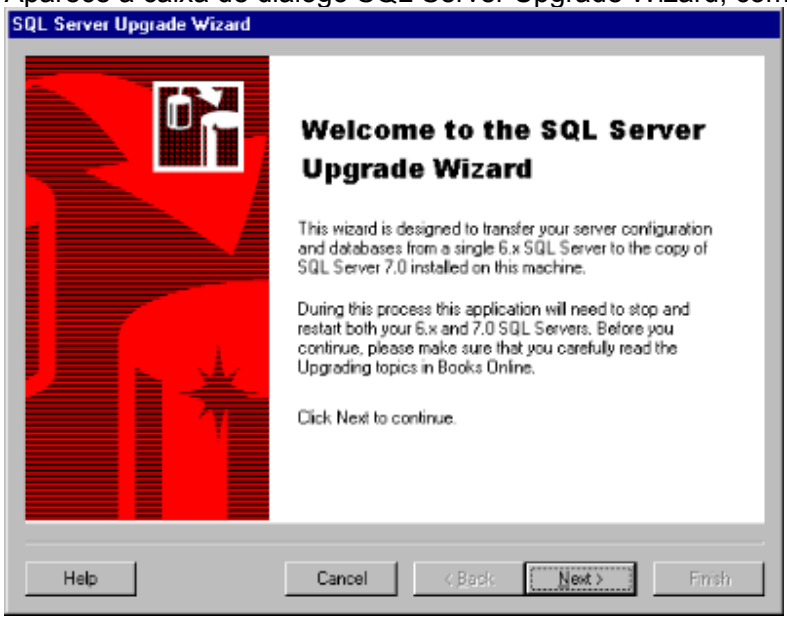
Iniciando o Asistente de Atualização de Versão



Do Menu Iniciar, selecione Microsoft SQL Server Switch e então selecione SQL Server Upgrade Wizard, como na figura abaixo:

:

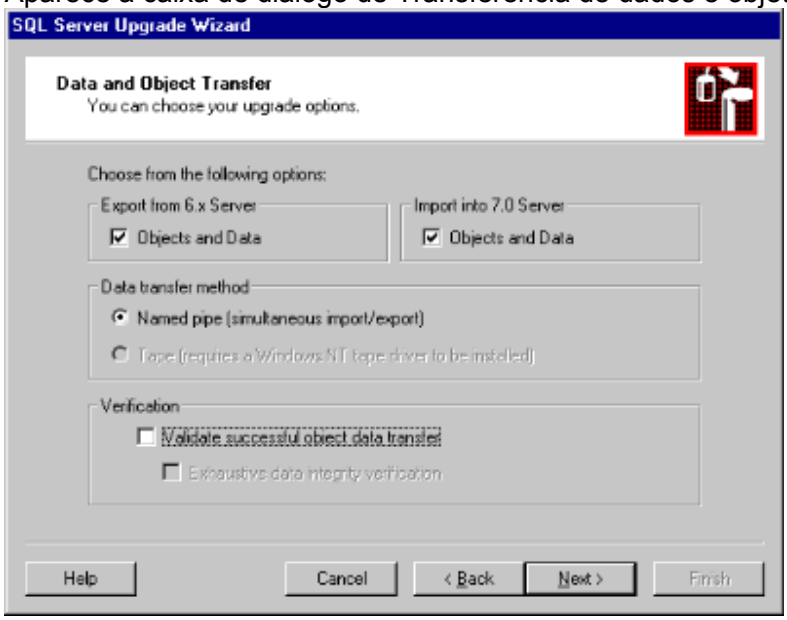
Aparece a caixa de diálogo SQL Server Upgrade Wizard, como mostrado abaixo:



Essa é apenas uma tela de boas vindas. Clique em Next para continuar.

Selecione a opção de transferência dos objetos.

Aparece a caixa de diálogo de Transferência de dados e objetos.



Nesta caixa de diálogo, você pode determinar o que você quer exportar do SQL Server 6.5 e o que você quer importar para o SQL Server 7.0. Os valores padrão são para exportar e importar dados e objetos, usar Pipes Nomeados (Named Pipes) para a transferênca de dados (esse é o Direct Pipeline). Pode-se mudar quaisquer das seleções padrão. Se a opção Named Pipe estiver selecionada (a opção Tape só é selecionável quando se tem uma unidade de fita no computador), não se pode desmarcar as caixas "Export from 6.x Server" nem a "Import into 7.0 Server", pois o método de Pipeline direto (Named Pipe) faz a exportação e importação simultânea.
A opção de Verificação, permite que se valide os dados depois da transferência (Validate successful object data transfer). Quando esta opção é selecionada, pode-se pedir que seja feita uma verificação de CRC (quase byte a byte) nos dados (marcando a opção Exhaustive data integrity verification). Quando ela é selecionada, aparece uma caixa de diálogo avisando que esta opção pode dobrar o tempo necessário para a atualização.
Abaixo estão os passos executados pelo Assistente de Atualização de Versão quando se escolhe um dos dois métodos (unidade de fita ou Pipe Nomeado).

Unidade de fita

1. Exporta os objetos 6.x
2. Fecha o SQL Server 6.x
3. Exprta os dados 6.x
4. Faz backup e copia os dispositivos (devices) 6.x
5. Inicia o SQL Server 7.0
6. Importa os objetos do SQL Server 6.x para o SQL Server 7.0
7. Importa os dados do SQL Server 6.x para o SQL Server 7.0

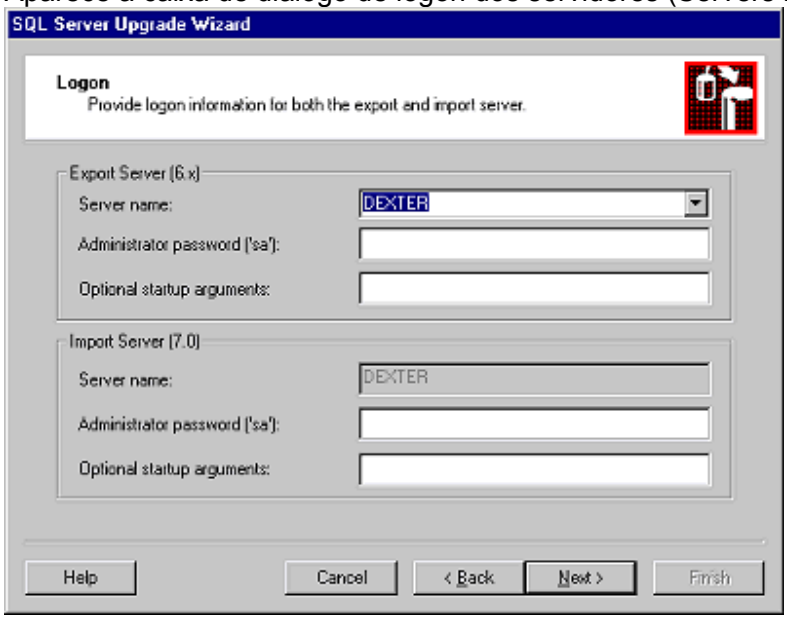
Direct Pipeline

1. Exporta os objetos 6.x
2. Fechao SQL Server 6.x
3. Iniica o SQL Server 7.0
4. Importa os objetos 6.x
5. Exporta e importa os dados do SQL Server 6.x para o SQL Server 7.0

Nota: Quando se seleciona a o método de transferência com a unidade de fita, você tem a a opção de copiar os dispositivos do SQL Server 6.x para uma uniadde de rede.

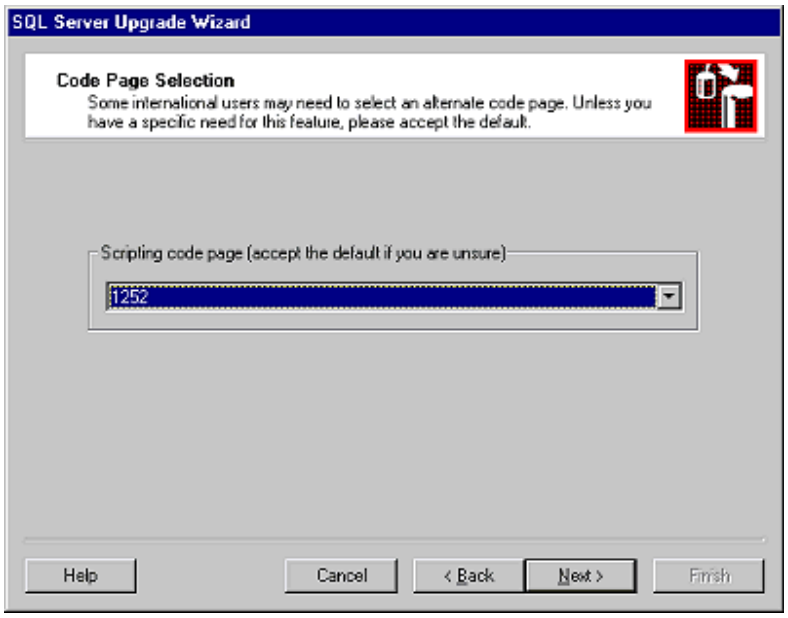
Logon dos servidores

Aparece a caixa de diálogo de logon dos servidores (Servers Logon).



Selecione em Server Name, o servidor 6.x de quem você quer exportar os dados. Entre com nome e senha do administrador (conta "sa"), e quaisquer opções de linha de comando adicionais necessárias para iniciar o servidor. O mesmo deve ser feito para o servidor 7.0 que você quer importar os dados para ele. Digite nome e senha do administrador, e parâmetros adicionais para iniciar o servidor.
Quando se clica em Next, aparece uma caixa de diálogo dizendo que será parado o SQL Server 7.0 e iniciado o SQL Server 6.5 e que ninguém pode estar acessando os servidores (o SQL Server 6.5 e o 7.0), e se você tem certeza que quer continuar. Caso você tenha certeza que não há ninguém acessando algum dos servidores, clique em Yes. Aparece então uma janela indicando que o SQL Server 7.0 está sendo finalizado e o SQL Server 7.0 iniciado.

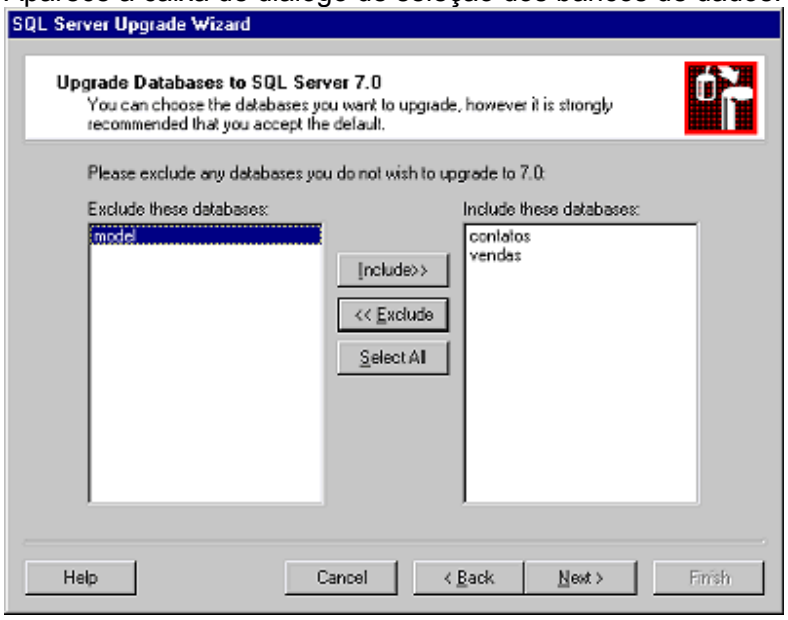
Seleção de página de código



A caixa de diálogo de seleção da página de código te permite escolher a página de código usada pra gerar os arquivos de script usados na atualização. Recomenda-se a opção padrão. Clique em Next para continuar.

Selecione os bancos de dados a atualizar

Aparece a caixa de diálogo de seleção dos bancos de dados.

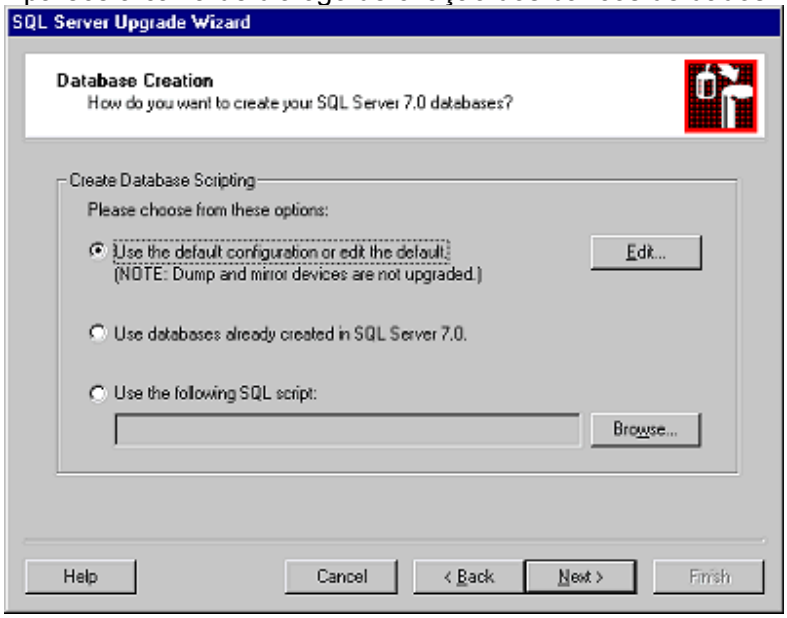


Selecione quais bancos de dados você quer atualizar colocando-os do lado "Include these databases". Os que você não quer atualizar, deixe-os do lado "Exclude these databases". Caso você decida-se por não atualizar todos os bancos de dados, aparece uma mensagem dizendo

que é recomendável que se atualize todos os bancos de dados de uma vez, e se você quer de fato fazer assim. Caso queira, clique em Yes. Após selecionados os bancos de dados que se quer atualizar, clique em Next.

Criação dos bancos de dados

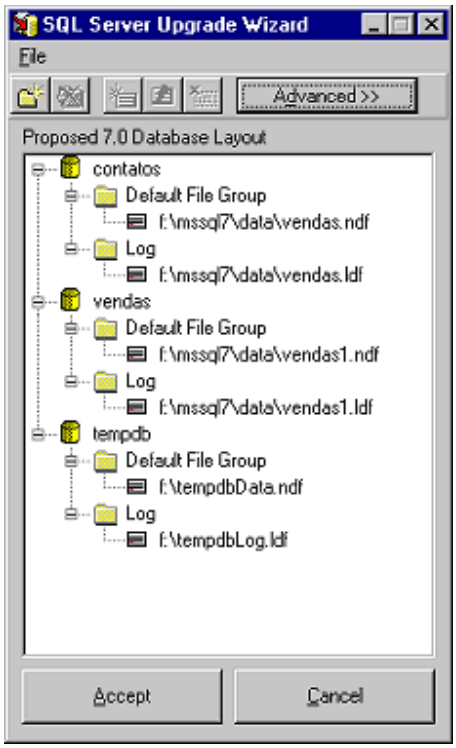
Aparece a caixa de diálogo de criação dos bancos de dados.

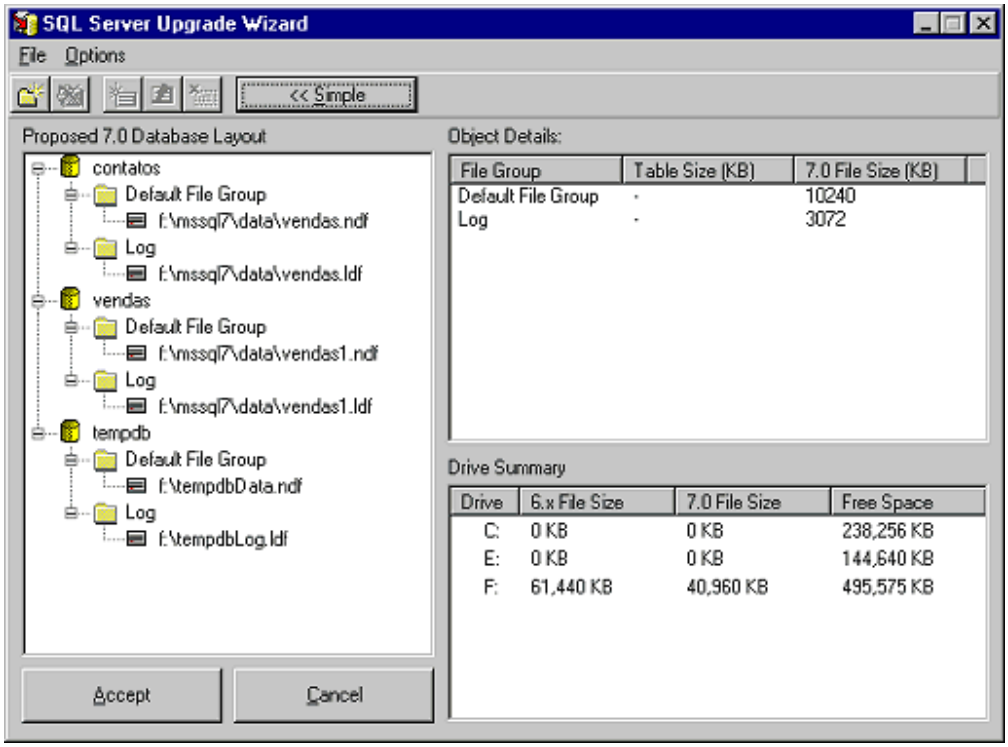


Para ter seus bancos de dados criados com as opções padrão, os arquivos de dados localizados no diretório de dados do SQL Server 7, e um mapeamento um-para-um dos dispositivos (devices) do SQL Server 6.5 para arquivos do SQL Server 7.0, use a configuração padrão. Para mudar arquivos ou locais, clique no botão Edit. Para usar bancos de dados já criados no 7.0, selecione a opção "Use databases already created in SQL Server 7.0". Para executar um script personalizado para criação dos bancos de dados, escolha a opção "Use the following SQL Script", e entre com o caminho e o nome do script a ser executado. Depois que você tiver feito suas escolhas, clique no botão Next.

Estimando o espaço em disco necessário

Pra estimar o espaço necessário para uma atualização por pipeline direto ou um atualização sem os bancos de dados do SQL Server, na janela mostrada na figura anterior, seleciona-se a opção Edit, depois Advanced. Aparece a janela "Proposed DataBase Layout".



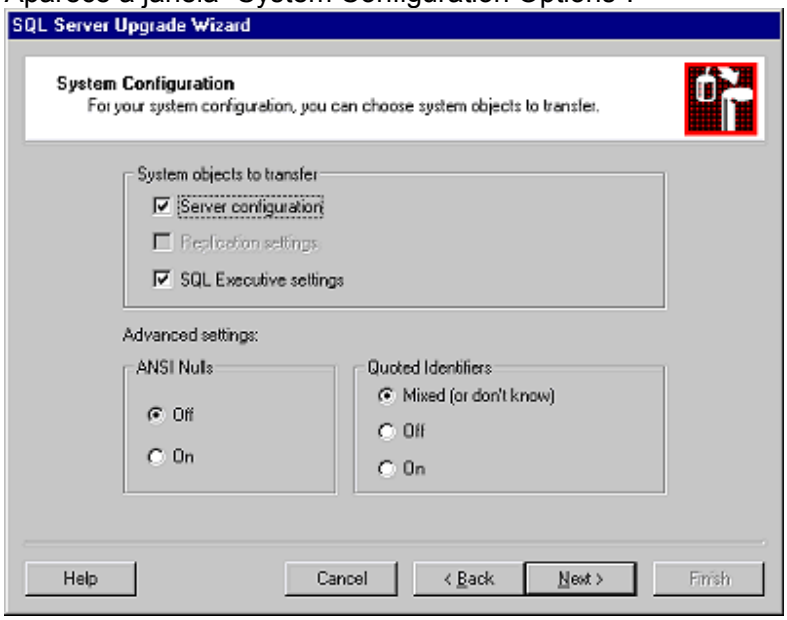


clique em Advanced e...

Nessa janela pode-se ver o espaço necessário para os bancos de dados 7.0 e o espaço livre no disco rígido. Para ver o espaço livre se for feita a remoção dos bancos de dados 6.5, selecionar "Options" e "Free Space Includes 6.x files". Nessa janela você pode definir outros arquivos para colocar os bancos de dados e, estando satisfeito com o definido clique em Accept. Aí volta-se para a tela de criação dos bancos de dados (Database creation). Clique em Next para continuar.

System configuration Options

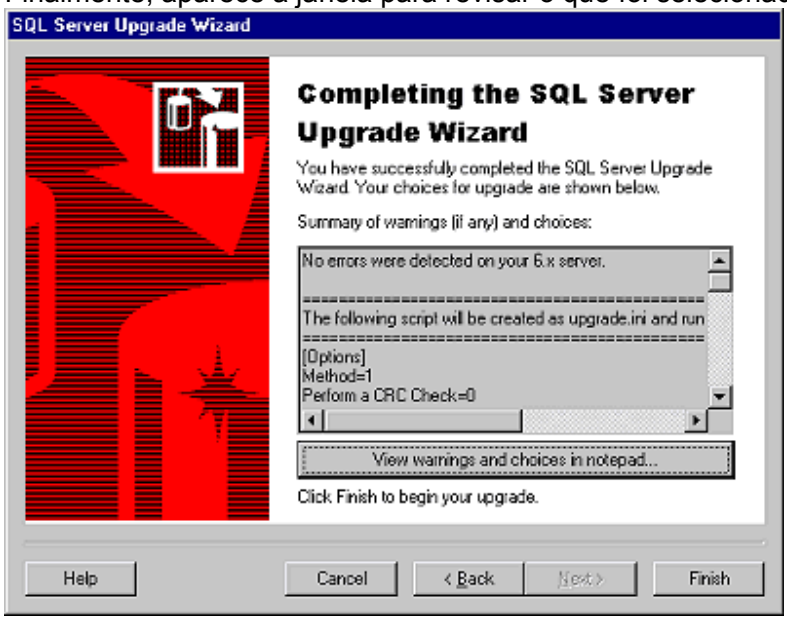
Aparece a janela "System Configuration Options".



Os valores padão são para transferir as configurações existentes de servidor e também as configurações do SQL Executivo, bem como desativar ANSI Nulls e ativar o modo misto de Quoted Identifiers. Selecionar as opções desejadas e clique em "Next".

Conferindo as seleções de atualização

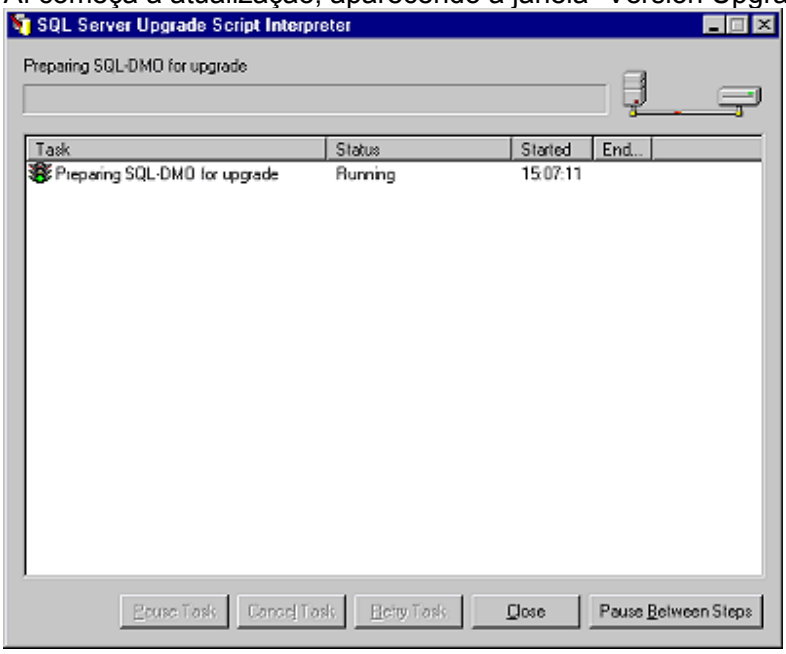
Finalmente, aparece a janela para revisar o que foi selecionado.



Aqui, você pode conferir todas suas seleções de opções de atualização e deve-se ajustar o que se quiser mudar até estar satisfeito com as seleções. Finalmente, clicar em "Finish" para começar a atualização.

Atualização para o SQL Server 7.0 em curso

Aí começa a atualização, aparecendo a janela "Version Upgrade Status".



Essa janela nos dá informações sobre o andamento do processo de atualização, como situação da tarefa, a tarefa sendo executada, e hora de início e término de cada uma das tarefas envolvidas na atualizaçao. Nessa janela, podemos pausar qualquer tarefa, (selecionando Pause Task), pausar entre tarefas (selecionando Pause Between Steps), abortar o processo de atualização (Close), abortar uma tarefa específica (Cancel Task), ou concluir uma tarefa pausada. Quando a atualização se completar com sucesso, aparece um aviso "Upgrade Complete".