

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 possui 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 índice Clustered a sua tabela será criada com o índice Non-clustered (ver abaixo).

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 especificado ele será criado como um índice Nonclustered.

É recomendado utilizar índices não agrupados [Nonclustered] para:

- Colunas que são usadas nas cláusulas ORDER BY e GROUP BY.

- Colunas que são frequentemente utilizadas como condições na cláusula 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.

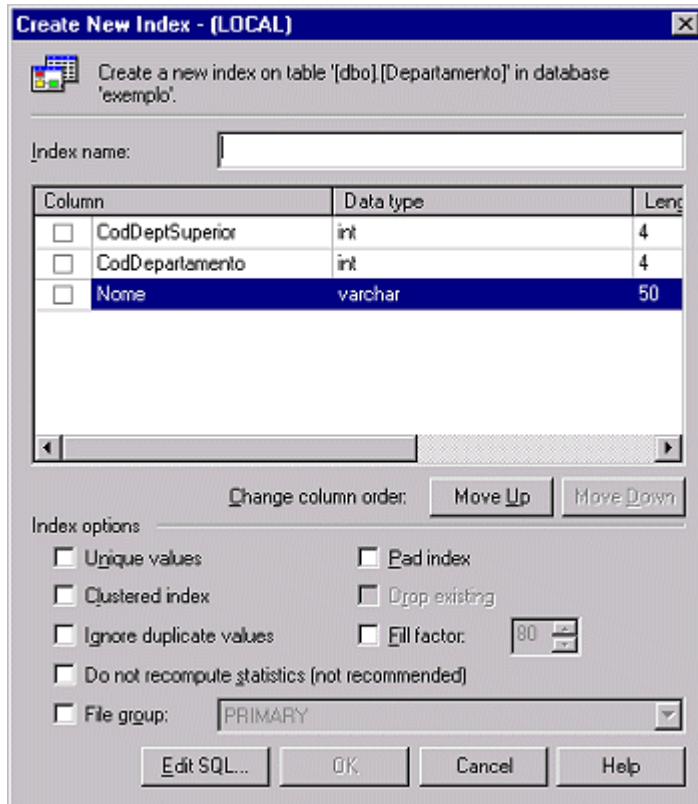
Se você criar um índice composto, o otimizador vai utilizá-lo mesmo quando apenas a primeira coluna é especificada, por exemplo em:

```
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.

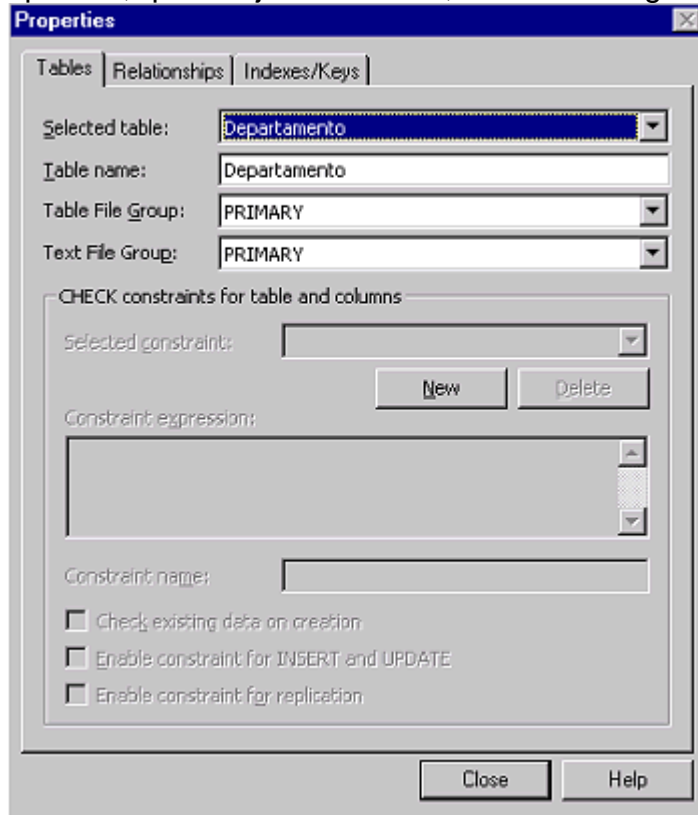
Nota: Um índice único não pode ser criado em uma coluna que já tenha valores duplicados, mesmo que você selecione a opção de "ignorar valores duplicados". Se você tentar, o SQL Server mostra uma mensagem de erro e lista os valores duplicados. Elimine os valores duplicados antes de criar um índice único 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 estatísticas 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 que duas.

Definidas todas as opções desejadas, clique em Ok, e o índice será criado (ou alterado). Você pode ver e alterar o código SQL usado na criação do índice, bastando para isso clicar no botão "Edit SQL". Quando editando o código, você pode verificá-lo antes de executar, bastando para isso clicar em "Parse". Clicar em "Execute", tem o mesmo efeito de clicar em Ok na tela anterior.

Para excluir um índice qualquer, pelo Enterprise Manager, basta selecioná-lo, na tela "Manage Indexes", e clicar em Delete.

Para criar um índice na criação de uma tabela basta , no modo de edição da tabela, selecionar qualquer coluna, clicar com o botão direito, e em Properties, na janela que aparece, que é a janela abaixo, selecionar a guia Indexes/Keys:



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 for 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 preenchimento, 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 opção, 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.índice 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:

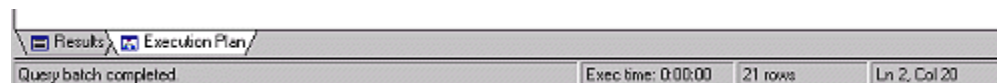
- Estimativa aproximada de quanto trabalho é necessário para usar o índice ou não. Essa estimativa é baseada em informações estatísticas sobre o índice, que dizem qual a distribuição dos dados. Essas informações podem ficar

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.

Analizando o Otimizador

 Para saber se o otimizador está usando seus índices ou não, no Query Analyzer, clique no botão "Show Execution Plan" (Ctrl+L). Quando da execução de uma



consulta, você agora tem duas guias, como pode ser visto abaixo:

Essas 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 processador 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 propriedade 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 mundo 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 criada 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 incompatível 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 valores 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_restricao* *definicao*

Onde a *definicao* inicia com as palavras PRIMARY KEY, UNIQUE, CHECK, FOREIGN KEY ou DEFAULT. A palavra CONSTRAINT e o *nome_da_restricao* 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 DefPaís 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_restricao
```

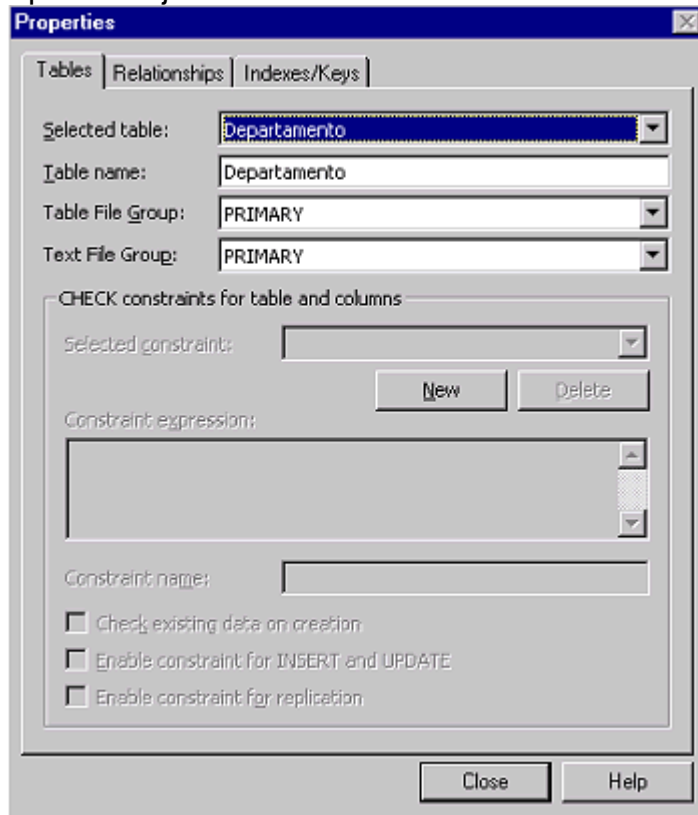
Uma restrição também pode ser desabilitada temporariamente e depois reabilitada com o comando ALTER TABLE, usando as opções NOCHECK (para desabilitar) e CHECK (para habilitar). Isso não funciona com PRIMARY KEY, UNIQUE ou DEFAULT, apenas com as outras restrições. Isso pode ser útil para inserir dados que fujam aos valores impostos pelas restrições. A sintaxe é:

```
alter table nome_da_tabela nocheck constraint nome_da_restricao
```

```
alter table nome_da_tabela check constraint nome_da_restricao
```

Gerenciando restrições com o Enterprise Manager

Todas as operações sobre restrições que fizemos podem ser feitas através do Enterprise Manager. Clique com o botão direito numa tabela, clique em **Design Table**, selecione alguma coluna, clique na mesma com o botão direito e selecione 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]

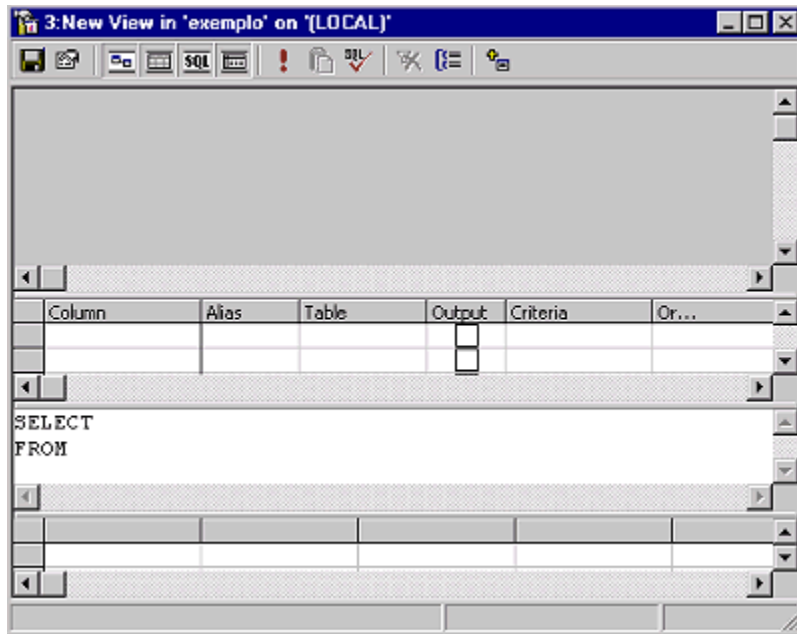
Uma visão [view] é uma forma alternativa de olhar os dados contidos em uma ou mais tabelas. Para definir uma visão, usa-se um comando SELECT que faz uma consulta sobre as tabelas. A visão aparece depois como se fosse uma tabela.

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 definição 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 recebe 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 execução 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 procedimento 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 proprietário 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 + '%'

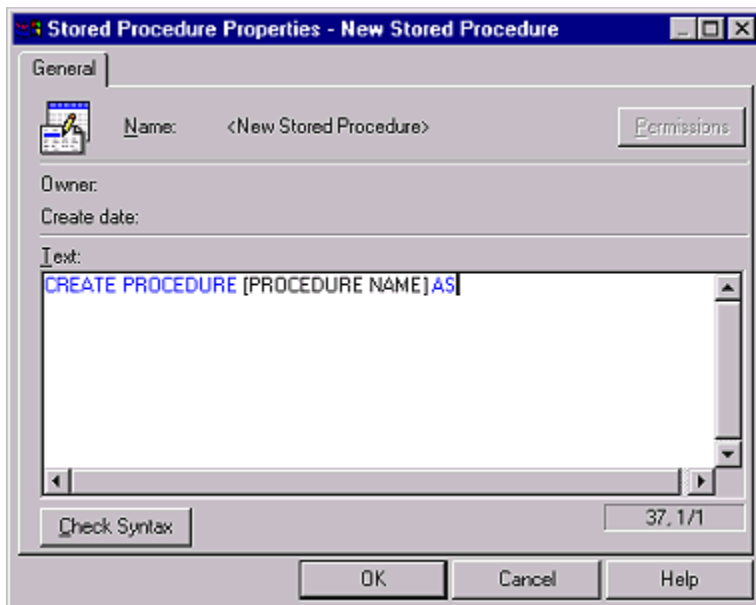
    if @contagem = 0
    begin
        select @mensagem = 'Nenhum cliente contém
'+@nomeBusca+' '
        print @mensagem
        print " "
    end

```

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 Manager. Para isso, deve-se expandir um grupo de servidores, 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 consequência. 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 valores 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 Authentication 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

Recursos avançados de segurança
Adicionar grupos como uma conta.
Acesso rápido.

Modo de segurança mista

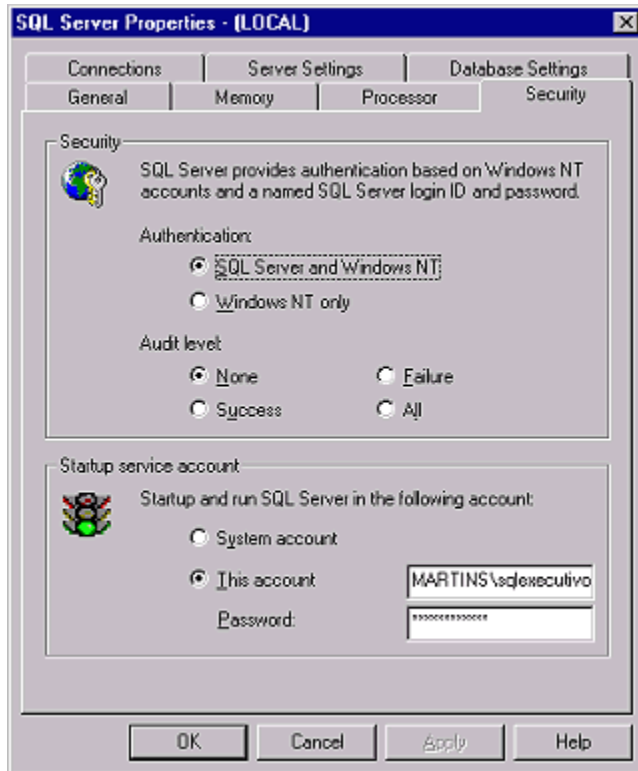
Clientes não-Windows e usando browser podem usar esse modo para conectar-se.

Camada adicional de segurança sobre o Windows NT

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

O SQL Server 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 desses papéis predefinidos de servidor tem permissões específicas para gerenciar os servidores SQL Server e todos os bancos de dados gerenciados pelo SQL Server. As tarefas administrativas que cada login pode executar dependem apenas de qual papel predefinido de servidor a que ele pertença. Os papéis predefinidos de servidor são:

- Administradores de sistema [System Administrators] (sysadmin): Este é o mais poderoso de todos os papéis. Qualquer um que pertença a esse papel pode realizar qualquer tarefa no SQL Server, inclusive sobrepor-se a qualquer dos outros papéis predefinidos de servidor. Esse papel é o equivalente à conta SA em versões anteriores do SQL Server (a conta SA, por padrão faz parte desse grupo).
- Criadores de bancos de dados [Database Creators] (dbcreator): Eles têm a habilidade de criar e alterar bancos de dados individuais.
- Administradores de discos [Disk Administrators] (diskadmin): Têm a capacidade de gerenciar arquivos de disco.
- Administradores de processos [Process Administrators] (processadmin): Têm a capacidade de gerenciar os vários processos sendo executados no SQL Server.
- Administradores de segurança [Security Administrators] (securityadmin): Eles têm a capacidade de gerenciar logins para um servidor.
- Administradores de servidor [Server 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 semelhante ao grupo Todos [Everyone] do Windows NT Server. Você não pode adicionar ou remover usuários deste papel, ou modificá-lo de qualquer maneira. Tudo que pode ser feito é atribuir permissões a ele. Quaisquer permissões atribuídas ao papel público são automaticamente atribuídas a todos IDs de usuário no banco de dados. O papel público é especialmente útil se você quiser atribuir as mesmas permissões para todos os usuários de banco de dados em um banco de dados, ao mesmo tempo.

Papéis personalizados de banco de dados [Custom Database Roles]

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

Se você estiver utilizando a autenticação do Windows NT e usa grupos globais do NT Server para gerenciar usuários, você perceberá que você não precisa realmente de criar papéis personalizados de banco de dados, já que você obtém o mesmo 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:

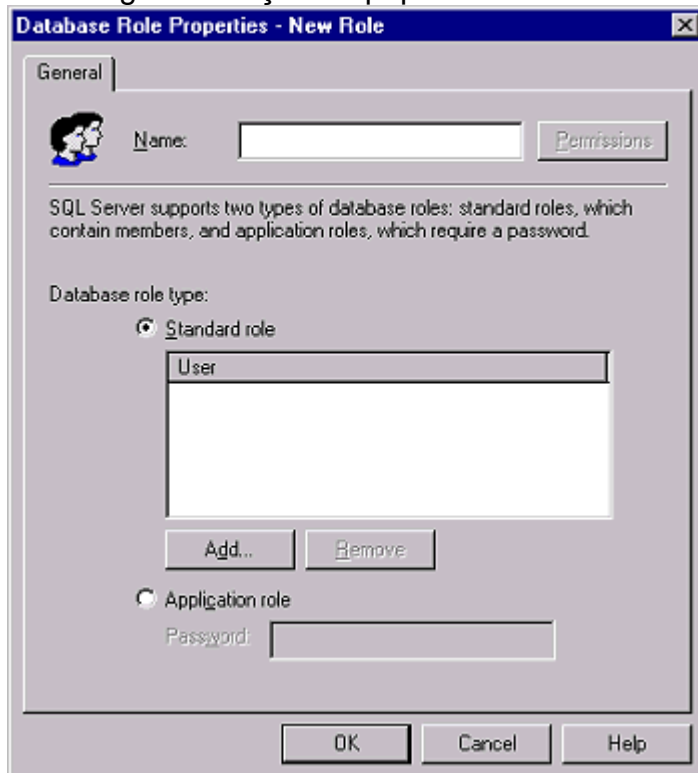
- Papéis personalizados de banco de dados, como ocorre com qualquer papel do SQL Server, são utilizados para agrupar usuários semelhantes que precisam do mesmo conjunto de permissões para acessar o SQL Server.
- Papéis personalizados de bancos de dados são criados dentro de um banco de dados e não podem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

Nota: Se você tiver permissão para a criação de grupos globais e utilizar a autenticação do Windows NT, você deve sempre usar grupos globais ao invés de papéis personalizados de banco de dados. O uso de grupos globais ao invés de papéis personalizados de banco de dados geralmente reduz o tempo necessário para

gerenciar as contas de usuário do SQL Server e do NT Server, pois grupos globais funcionam com ambos. Papéis personalizados de banco de dados funcionam apenas no SQL Server. Além disso, grupos globais podem se estender a vários bancos de dados, enquanto papéis são específicos para cada banco de dados, o que os torna menos flexíveis que grupos globais.

Criando e configurando papéis de banco de dados

Veremos agora como criar um papel personalizado de banco de dados. Para isso, no 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 (clcando 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 deve 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 frequência, 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:

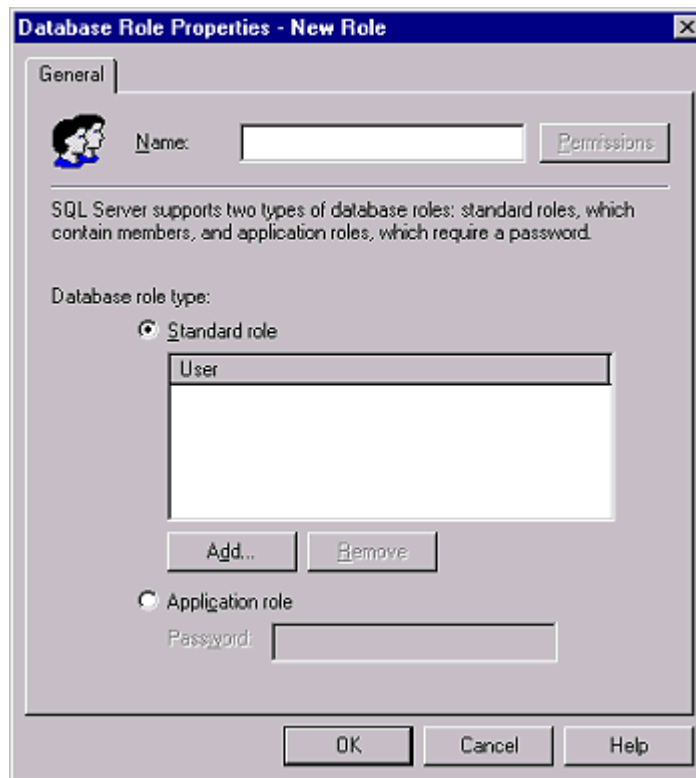
1. No Enterprise Manager, expanda o banco de dados em que está definido o papel que você quer excluir.
2. Clique em Roles e, no lado direito da tela, selecione o papel a ser excluído. Dê um duplo clique no mesmo, e verifique se em baixo de **User**, há algum usuário listado.
3. Caso haja algum usuário, 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 login de um papel de servidor, faça o seguinte:

- No Enterprise Manager, selecione o servidor SQL Server cujos papéis você quer configurar. Expanda-o e abra a pasta **Security**.
- Clique em **Server Roles**. No lado direito da tela aparecem os papéis de servidor. Selecione o papel ao qual você quer adicionar algum login.
- Clique no mesmo com o botão direito e selecione Properties. Aparece a janela abaixo:
- Para adicionar um login ao papel de servidor, clique no botão **Add**. Aparece a caixa de diálogo "Adicionar Membros" [Add Members], com todos os logins definidos para o servidor.
- Escolha um ou mais logins para adicionar a esse papel de servidor. Cada vez que você selecionar um login, ele ficará marcado, e assim ficará até que você o clique de novo. Depois de selecionados todos os logins que você quer adicionados ao papel de servidor, clique em Ok. Então você volta para a caixa de diálogo de propriedades do papel de servidor (mostrada acima).
- Caso você queira remover algum login que faz parte de um papel de servidor, selecione-o, na caixa de diálogo de propriedades do papel de servidor, e clique no botão **Remove**.
- Quando você tiver adicionado e/ou removido todos os logins desejados a esse papel de servidor, clique em Ok para concluir.

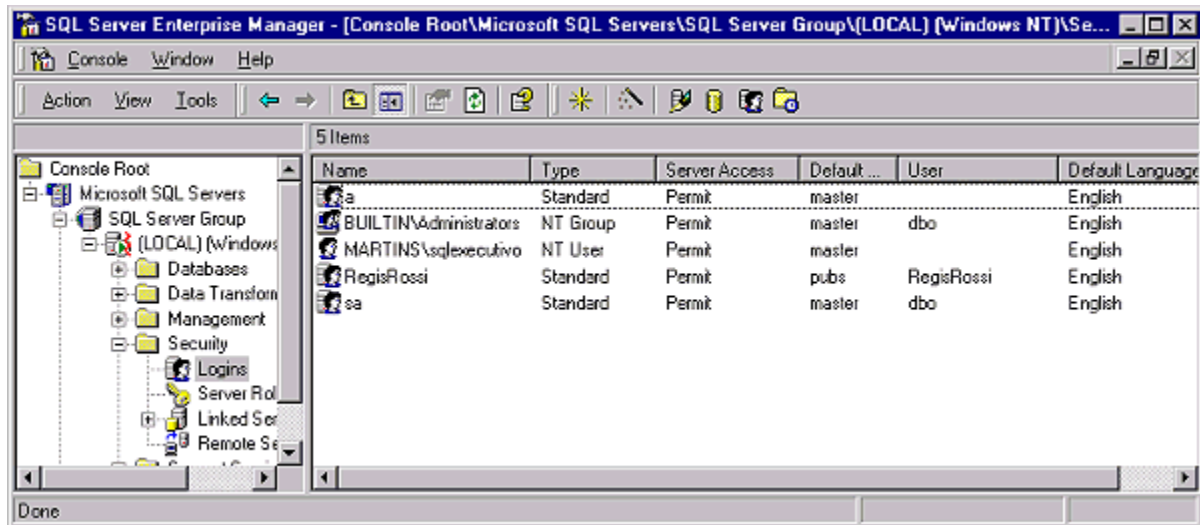
Você pode adicionar logins aos papéis de servidor sempre que achar necessário. Mas lembre-se que o ato de delegar privilégios administrativos a usuários às vezes pode ser

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

Visualizando informações de segurança

Visualizando informações de logins do SQL Server

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

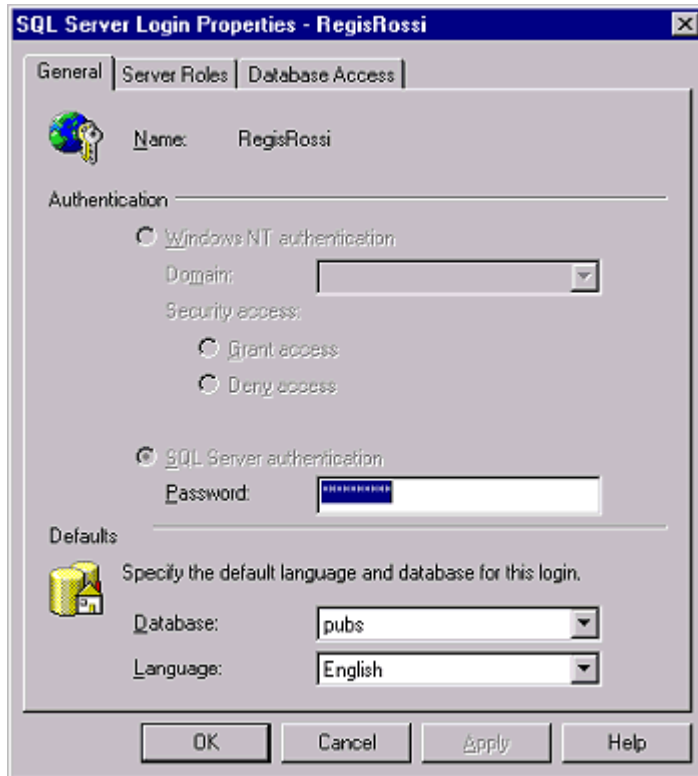


Observe na parte direita da tela estas informações:

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

- A coluna "Default Language" mostra a língua específica para o login. O padrão é inglês [English].

Para ver informações específicas sobre qualquer um dos logins, clique no mesmo com



o botão direito, e selecione Properties. Aparece a tela abaixo:

Aqui você pode ver e configurar quase todas opções de login. Essa tela tem três guias. Na guia General, você pode alterar a senha para esse login [Password], e definir seu banco de dados e linguagem padrão ([Database] e [Language]).

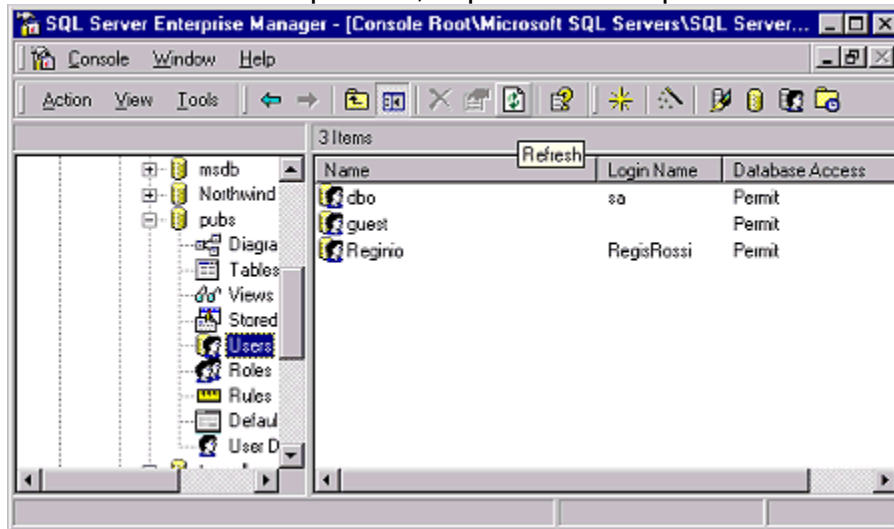
Na guia "Server Roles" mostra a quais papéis de servidor o login pertence. A guia "Database Access" mostra a quais bancos de dados o login tem acesso (ou seja, tem um login definido na tabela *syslogins* do banco de dados), além de mostrar a quais papéis de banco de dados o usuário pertence, em cada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

Clique em Cancel para sair da janelinha 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 Access" 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.

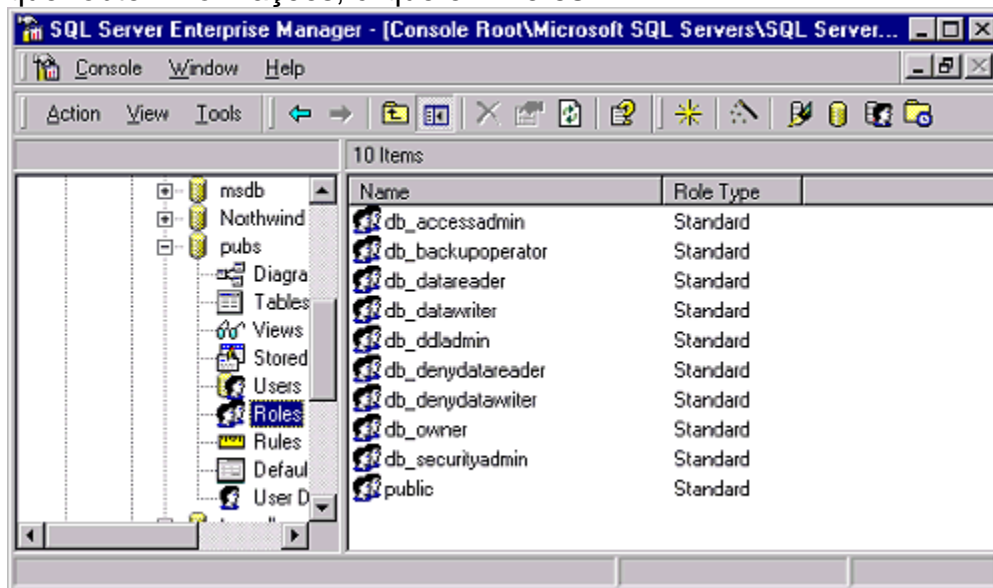
Essa tela mostra todos os papéis de banco de dados definidos para este banco de dados (todos que aparecem listados) e também a quais deles este usuário específico pertence (os que têm a caixa de verificação ao seu lado marcada).

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.

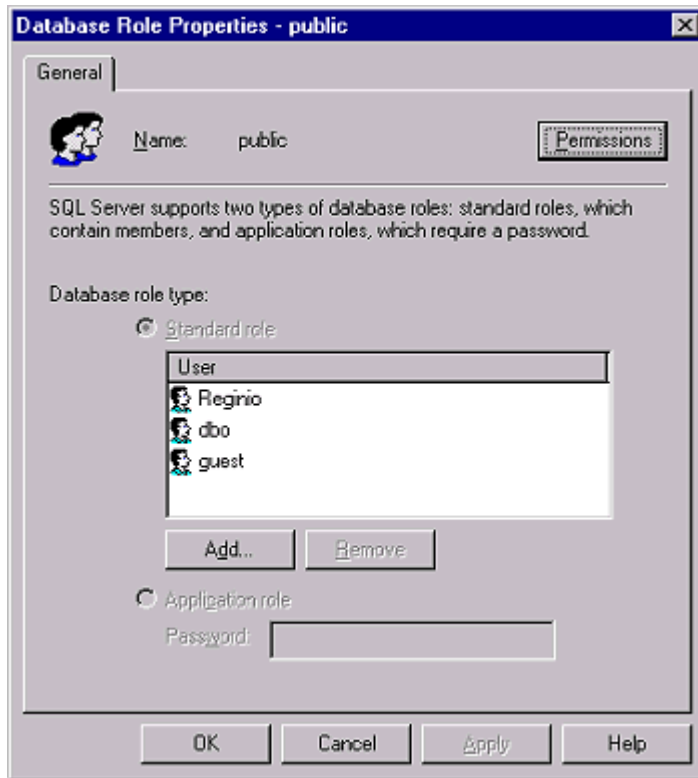
Para isso, no Enterprise Manager, expanda o banco de dados de cujos papéis você quer obter informações, clique em **Roles**.



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, enquanto 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.

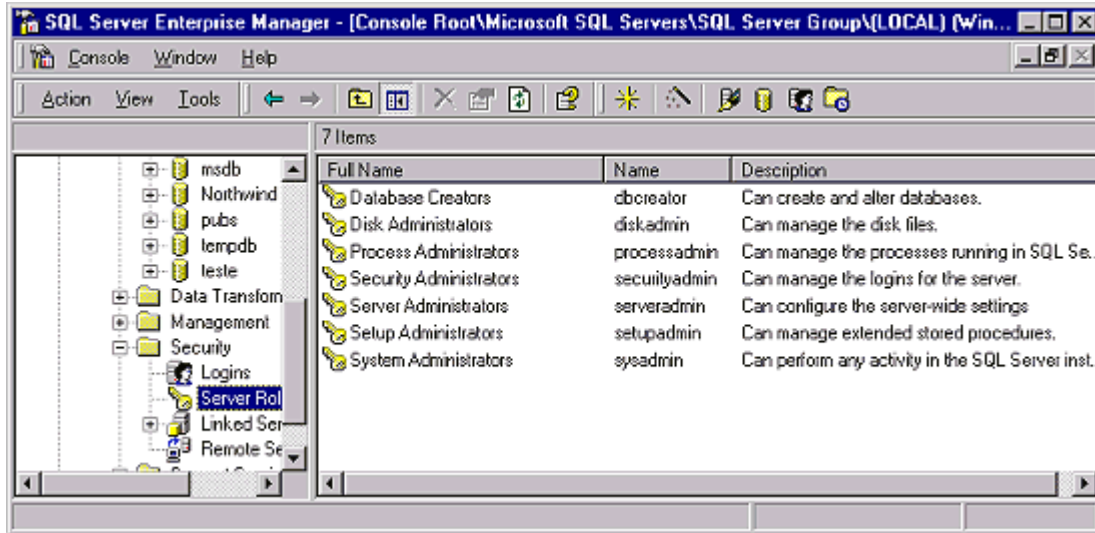
Para sair dessa caixa de diálogo, clique em Cancel.

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

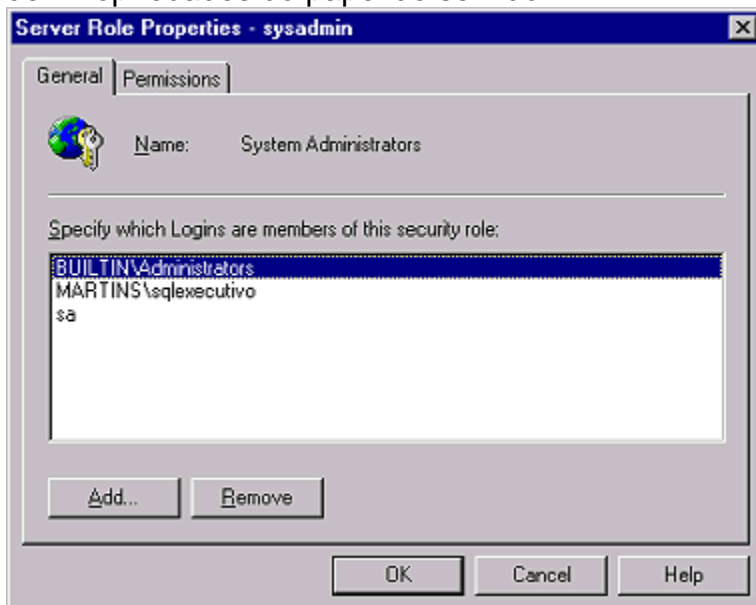
Muitas vezes, você irá querer ver os vários papéis de servidor de seu SQL Server e determinar quais logins pertencem a quais papéis de servidor.

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

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

Permissões implícitas

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

Permissões implícitas podem assim ser atribuídas a usuários, papéis personalizados ou grupos, com a simples atribuição dos mesmos a um papel predefinido de banco de dados ou de servidor. As permissões implícitas também podem ser atribuídas a usuários, grupos ou papéis personalizados definindo quaisquer destes como o proprietário de um objeto de banco de dados específico.

Os usuários, grupos ou papéis personalizados podem ser atribuídos a qualquer um dos

Papéis predefinidos de Servidor ou **Papéis predefinidos de Banco de Dados**, recebendo as permissões que tal papel tenha. (relembre quais são os Papéis predefinidos de Servidor e Papéis predefinidos de Banco de Dados)

Além de receberem permissões através da atribuição aos papéis acima, também podemos fazer usuários tornarem-se proprietários de algum objeto. Como funciona isso?

Quando um usuário com a permissão de comando adequada cria um novo objeto no banco de dados, tal como uma tabela, ele se torna o *proprietário do objeto de banco de dados* [Database object owner] (DBOO) daquele objeto. Proprietários de objetos de banco de dados têm permissões implícitas em todos os objetos que lhes pertençam, o que os dá a capacidade de executar qualquer atividade naquele objeto, tal como SELECT, INSERT, UPDATE, DELETE, entre outros. Eles têm controle completo dos objetos que criam.

Como dá para perceber, permitir que qualquer um seja um DBOO não é uma boa idéia. Normalmente, as únicas pessoas que devem criar objetos de bancos de dados são DBAs ou desenvolvedores SQL, não usuários comuns.

Precedência de permissões

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

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

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

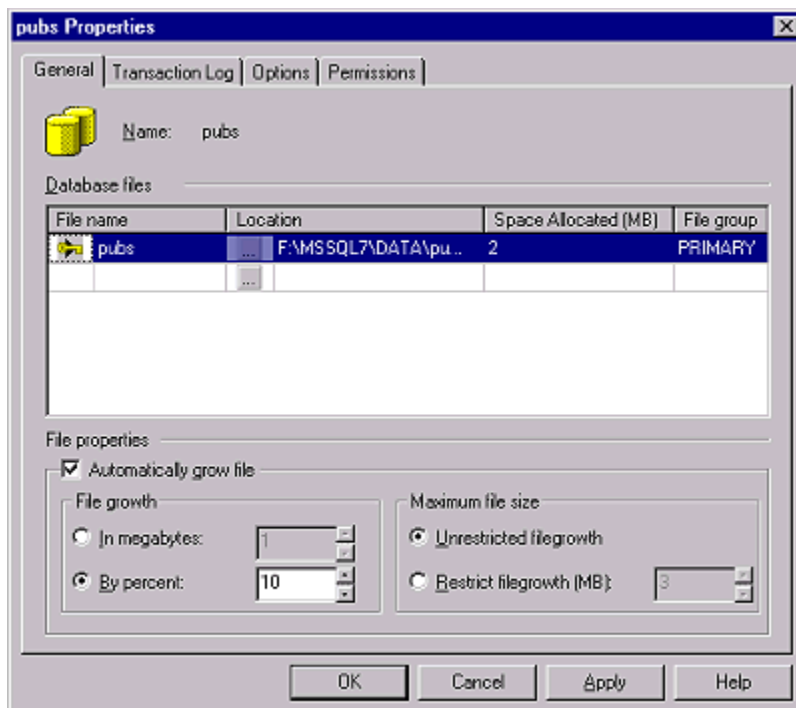
O que ocorre se um usuário receber permissões diferentes através de várias permissões individuais, vários grupos ou papéis de que o mesmo faça parte?

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

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

Visualizando informações de permissões

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

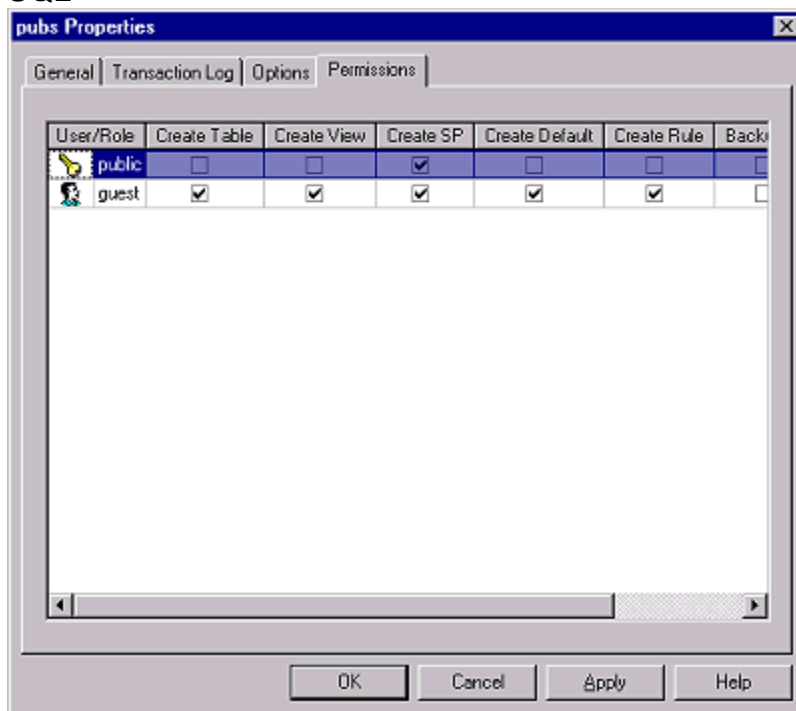


de comandos. Isso não apenas te ajudará a trabalhar com permissões, mas também te mostrará como executar uma tarefa que você estará realizando regularmente como um DBA. Vamos ver como visualizar as permissões atuais de objeto para todos os usuários, grupos, e papéis em um único banco de dados utilizando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

Visualizando permissões para comandos SQL

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

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



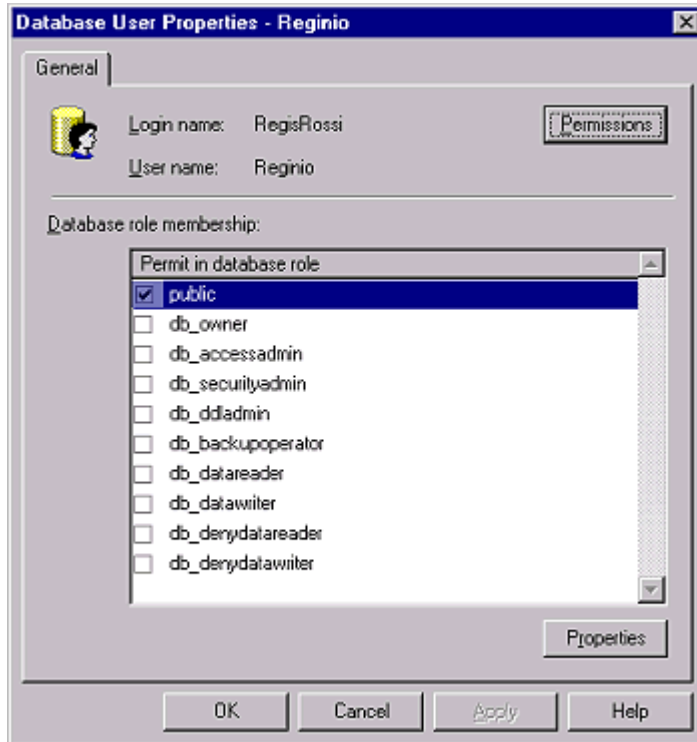
Na primeira coluna desta tela, embaixo do título **User/Role**, estão listados todos os IDs de usuários de bancos de dados para esse banco de dados. Lembre-se que essa coluna pode exibir quaisquer grupos, papéis ou usuários. Nas outras colunas estão as várias permissões para comandos SQL que podem ser 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

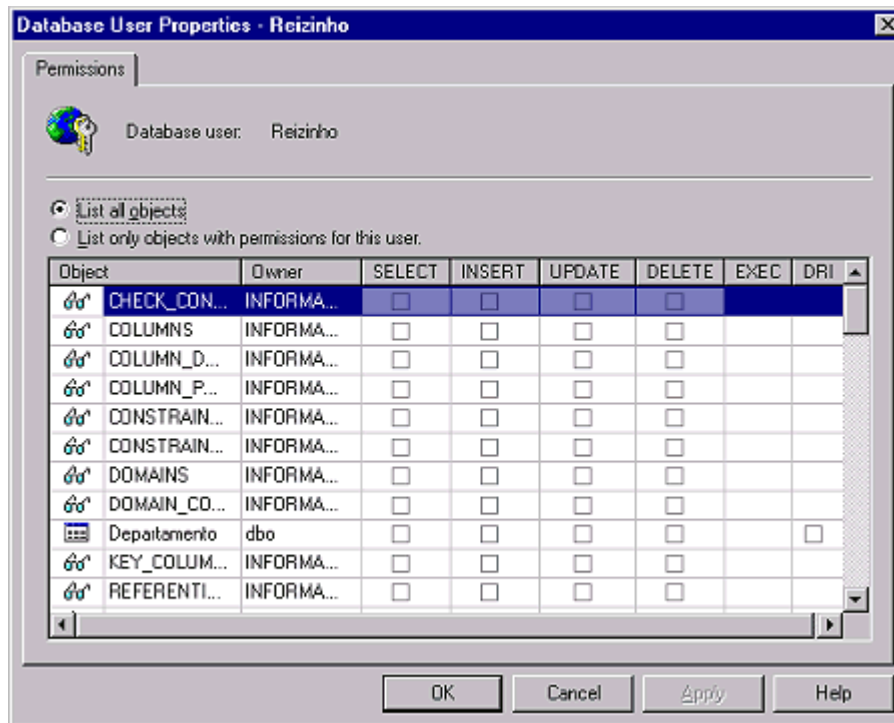
A visualização de permissões de objetos é um pouco mais difícil do que a visualização de permissões para comandos SQL. Você pode vê-las da perspectiva de usuários, grupos, ou papéis, ou então da perspectiva dos próprios objetos. Analisaremos aqui as duas maneiras.

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 tem 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.

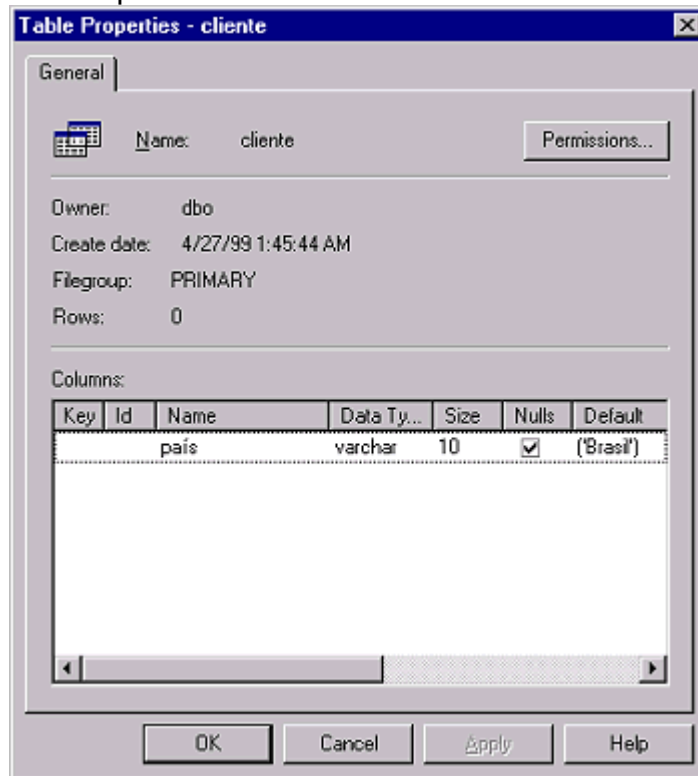
Perceba que nem todos os objetos têm todas as permissões de objeto disponíveis. Por exemplo, procedimentos armazenados têm apenas a permissão de objeto **Execute**.

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

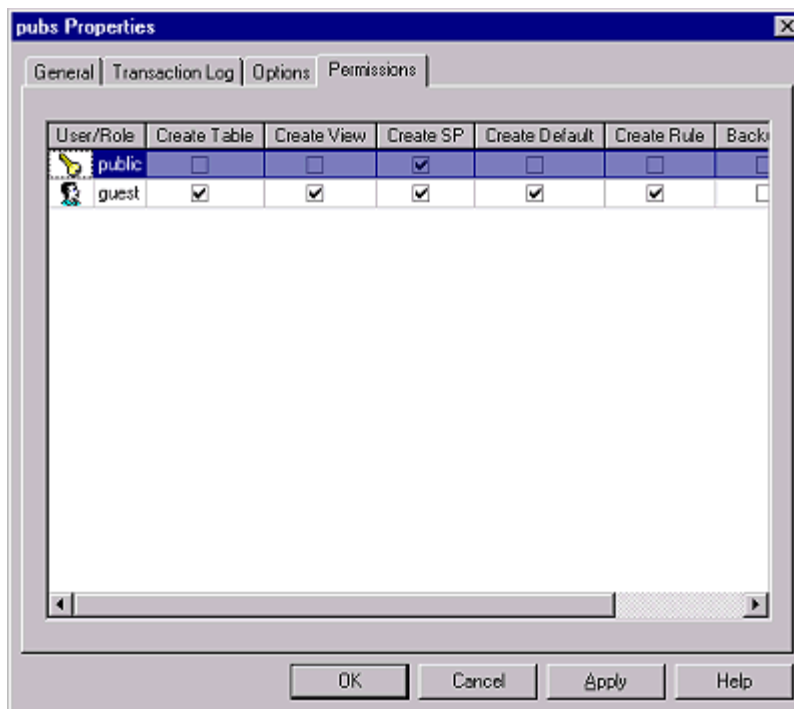
papel obteve a permissão de objeto associada. Veja que nem todos os objetos têm todas as permissões de objeto.

- 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, atribuiremos 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 ela 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.

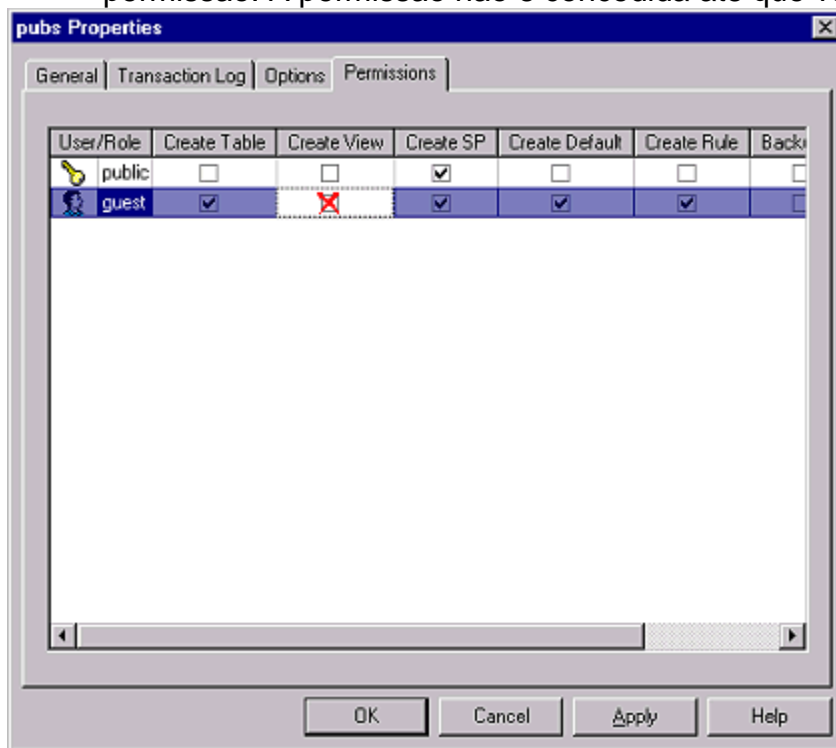
Para conceder ou revogar uma permissão utilizando o Enterprise Manager, os passos são os seguintes:



- 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:

- 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 usuário, 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.
- 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.



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

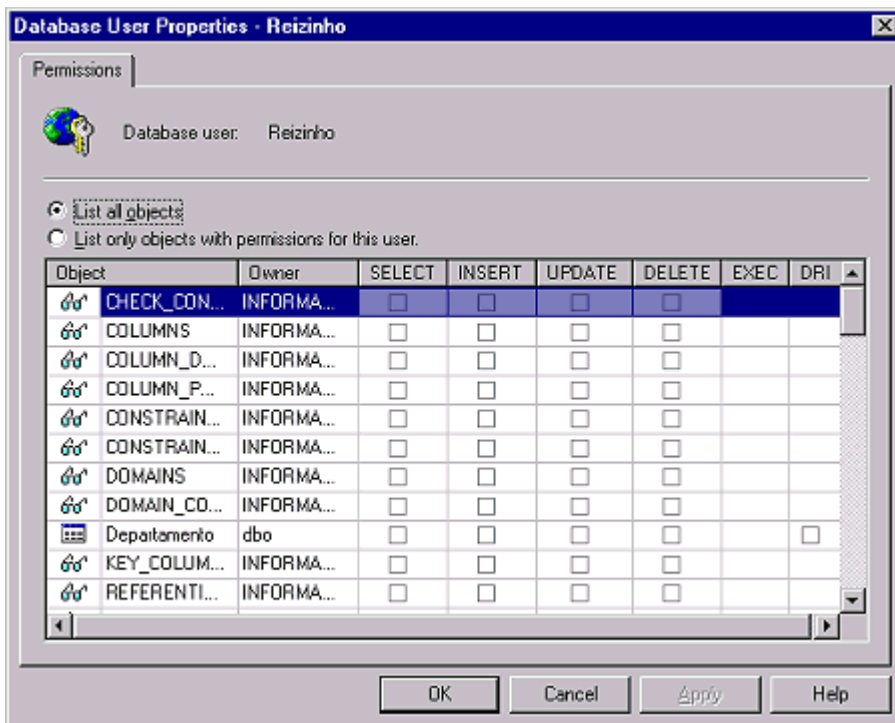
Concedendo e revogando permissões de objetos pelo Enterprise Manager

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 terminar 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, digitariamos 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 backup de todo banco de dados. Assim, em um banco de dados que é bastante modificado 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 informação 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 TRANSACTION, 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 backup pode especificar o nome de um arquivo em disco rígido que será escrito quando o backup 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.

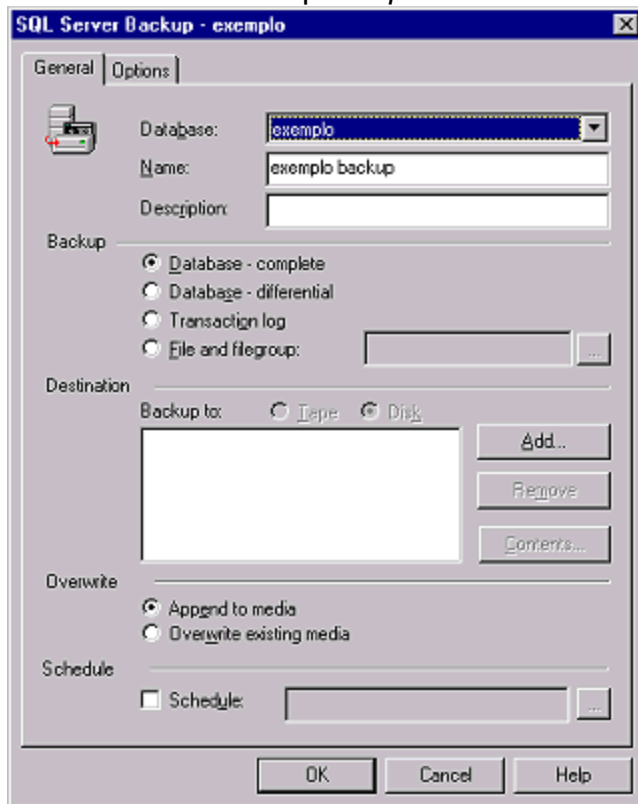
O Enterprise Manager cria o nome físico do arquivo em C:\MSSQL\BACKUP, como default, mas pode ser mudado. Se você clica em "Tape backup device", você deve escolher um dos dispositivos de fita suportados pelo Windows NT, como nome físico. No caso, usaremos "Disk backup device". Clique em Ok.

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 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 substitua, sobrescreva, o conteúdo da fita ou arquivo para onde ele está sendo gravado.
6. Clique em Ok para iniciar 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.



backups, selecione-o na lista de
que no mesmo e clique no botão
e inclui o banco de dados

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 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 significa 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 limpo frequentemente, seu espaço pode se esgotar. Isso pode acontecer caso não sejam feitos backups frequentes 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.

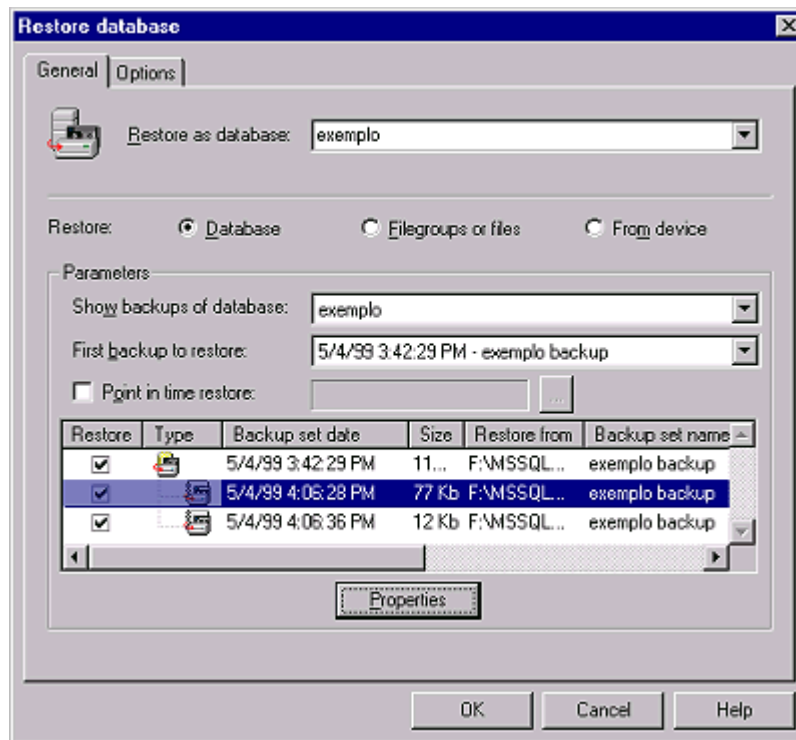
Restaurando um Backup

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 restaurar, 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, verá uma figura semelhante à número 2.

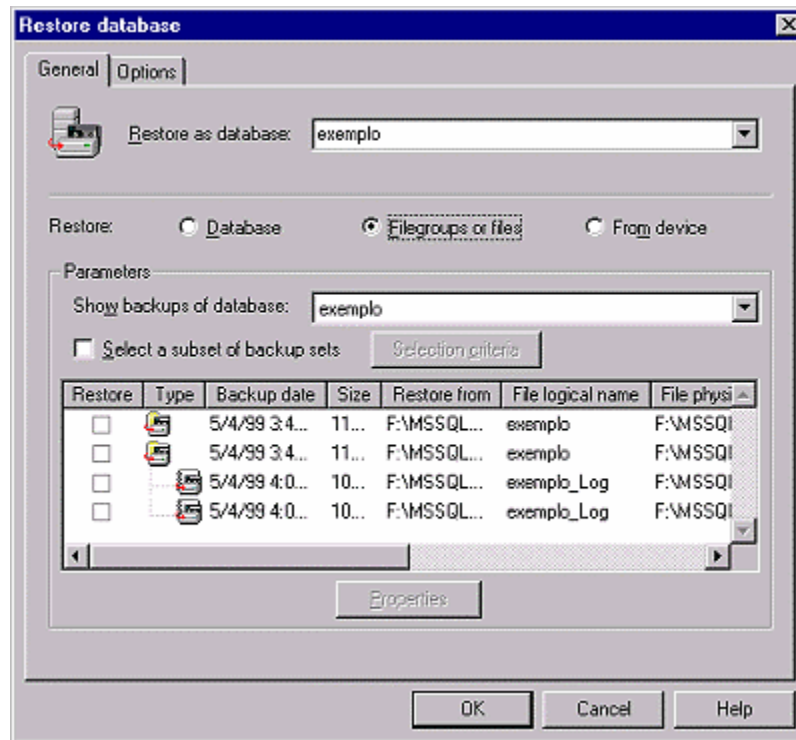


- Caso selecciones From Device, verá algo parecido com a figura número3.

1

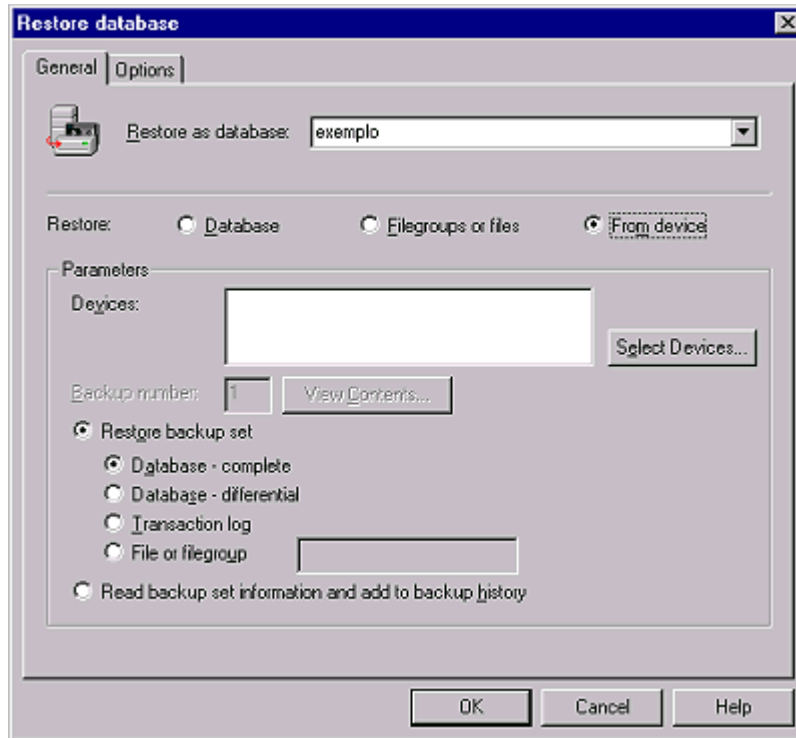
A primeira opção, "Restore Database", que mostra a figura acima, exibe uma lista dos backups a partir dos quais você pode fazer a restauração, baseado em tabelas de histórico armazenadas no SQL Server. Por padrão, o backup completo do banco de dados mais recente é mostrado primeiro, seguido pelos backups 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 restauração 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 possível;.

Se você não quiser restaurar também os backups do log de transações, você pode desseleccioná-los clicando na caixa de verificação à esquerda de cada log. Se você desseleccionar 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. Então 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.

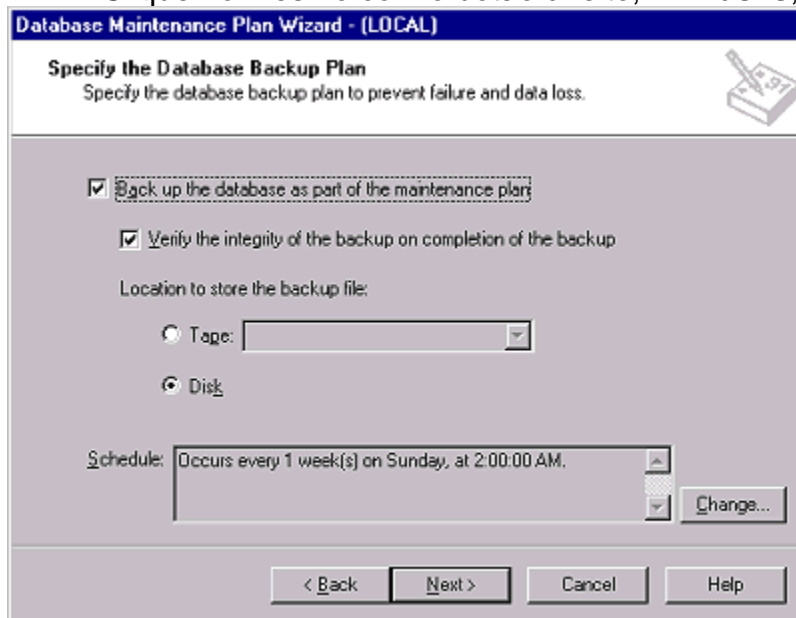
Há três meios de se criar um trabalho de backup:

- Utilizar o assistente de plano de manutenção de banco de dados [Database Maintenance 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 backup 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 oferece maior flexibilidade.

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

Para agendar um backup de banco de dados com o uso de um assistente, faça o seguinte:

1. No Enterprise Manager, selecione o banco de dados cujo backup você quer agendar.
2. Clique no mesmo com o botão direito, **All Tasks, Maintenance Plan**.



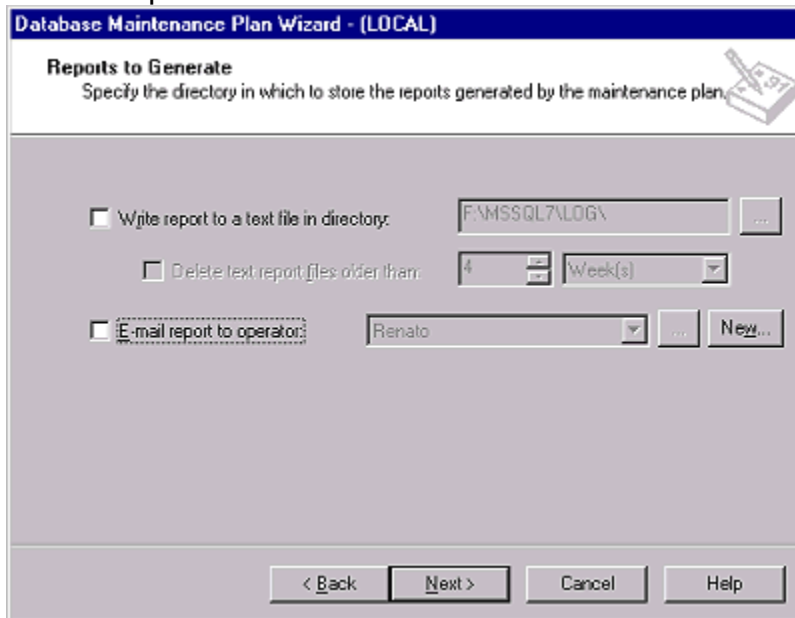
3. Clique em Next até chegar à tela chamada Especificar o plano de backup do banco de dados [Specify the Database Backup Plan], mostrada abaixo:

Você pode optar aqui por verificar a integridade do backup após seu término. Para isso, clique em "Verify the integrity of the backup on completion 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 for 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 maintenance 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 gerar [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 executando 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.

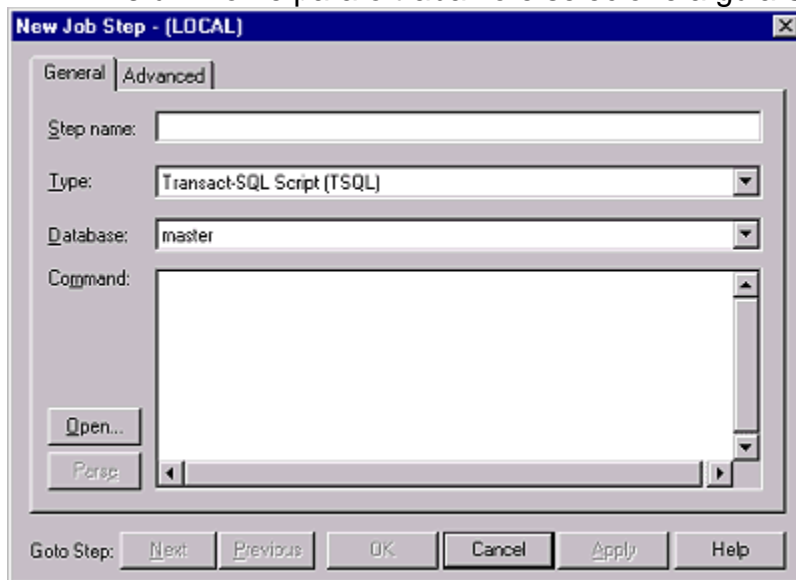
Como agendar um backup diferencial ou de grupo de arquivos

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 frequência o 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 botão 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 TO 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 grupo 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 opção "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 pode 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 memory 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 troca 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:

<i>name</i>	<i>varchar(30)</i>	O nome do objeto.
<i>id</i>	<i>int</i>	Um número de identificação. Pode ser obtido também com a função <i>OBJECT_ID(nome)</i> . 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)
<i>type</i>	<i>char(2)</i>	

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. Suas colunas mais importantes são:

<i>id</i>	<i>int</i>	Identificador da tabela ou do procedimento armazenado. Faz referência a <i>sysobjects.id</i> .
<i>colid</i>	<i>int</i>	Número seqüencial da coluna dentro da tabela (ou do parâmetro dentro do procedimento).
<i>type</i>	<i>int</i>	Identifica o tipo de dados básico, que faz referência a <i>systypes.type</i> .
<i>length</i>	<i>int</i>	Tamanho da coluna ou zero se não se aplica.
<i>usertype</i>	<i>int</i>	Tipo de dados definido pelo usuário. Faz referência a <i>systypes.usertype</i> .

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*. Essas 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:

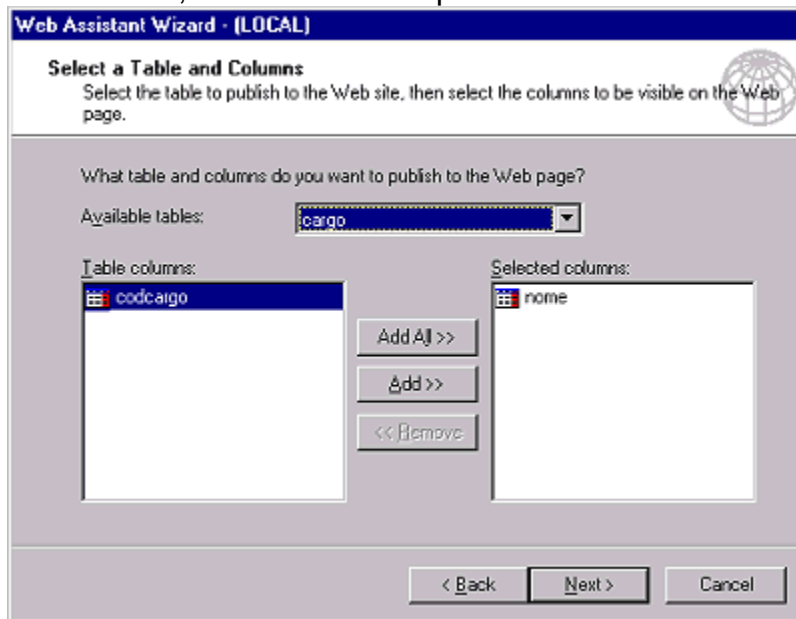
- "Data from the tables and columns I select": selecionando esta opção, lhe será permitido escolher quais colunas e tabelas fornecerão os dados para publicação. Caso você selecione esta opção siga para Seleção de tabelas e colunas.

- "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": obter 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.

Seleção 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:

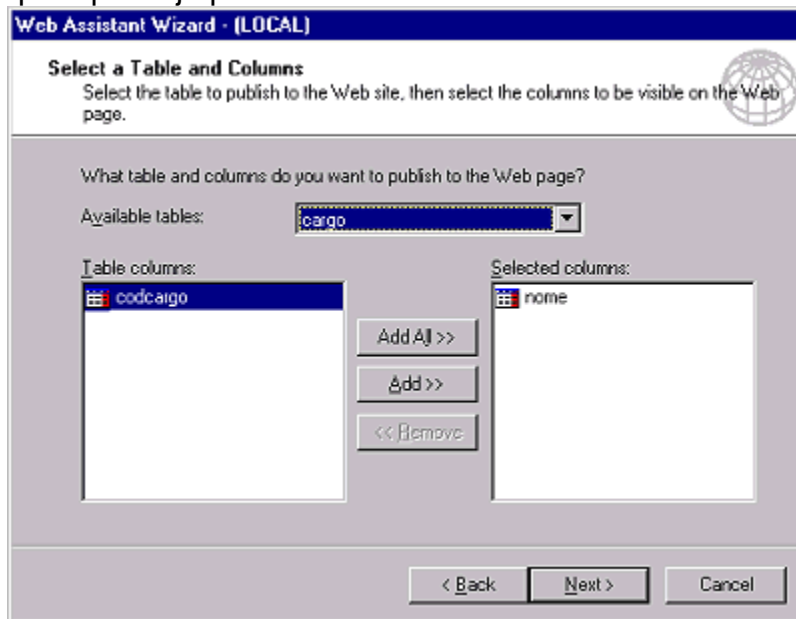
- All of the rows: mostrará todas as linhas da tabela que houver sido selecionada.
- "Only those rows that meet the following criteria": aqui você pode definir um critério para exibição das linhas, especificando uma coluna de uma tabela, um critério de comparação (=, <>, <, >, >=, <=), e um valor (número, 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.



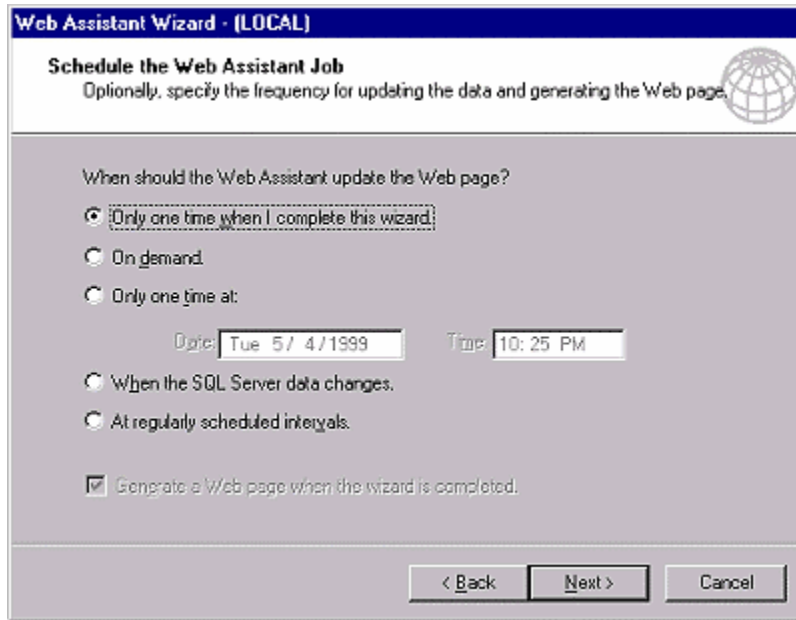
Clique em Next. Lhe será pedido para informar valores para todos os parâmetros necessários para o funcionamento do procedimento armazenado. Informe-os, clique em Next, e vá para Schedule the Web Assistant job.

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:

- | | |
|---|--|
| <i>On demand</i> | [Sob demanda]. O trabalho é executado mais tarde. Deve-se executar o Web Assistant de novo para executar a tarefa |
| <i>Only one time at:</i> | [Apenas uma vez em]. A página só é gerada uma única vez, no dia e hora especificados. |
| <i>When the SQL Server data changes</i> | [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. |
| <i>At regularly scheduled intervals</i> | [Em intervalos regularmente agendados]. Por exemplo, a cada 2 horas, a cada 2 dias etc. Clique em Next e, para esta opção, lhe será pedido 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 arquivo 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 arquivo 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 together.]. 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 definidas 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 seu "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).

Inserir 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, 'Abílio Frenkel', 'F', '222-2870', '031-295-4095', 'Av. Goiás,
345 sala 2', '', 'Goiânia', 'GO', '')
insert into Pessoa(CodPessoa, Nome, Sexo, Fone, Fax, Rua, Bairro, Cidade, Estado, CEP)
  values (2, 'Adão Dias', 'F', '261-8263', '222-4280', 'Rua C- 146, 661', '',
'Goiânia', 'GO', '')
insert into Pessoa(CodPessoa, Nome, Sexo, Fone, Fax, Rua, Bairro, Cidade, Estado, CEP)
  values (3, 'Adão Pereira', 'F', '234-7755', 'nôo lembra', 'Rua 2, 151',
'Setor Universitário', 'Goiânia', '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', '', 'Goiânia', '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', '', 'Goiânia', 'GO', '')

```

```
/**** Produto *****/insert into  
Produto(CodProduto, Nome, Descricao, QuantDisponivel, QuantMinima, Localizacao, Prec  
o) values(1, "Lâmpadas 150 V", "", 4.00, 2.00, "Estante , Prateleira , Divisão  
", 15.0000)insert into  
Produto(CodProduto, Nome, Descricao, QuantDisponivel, QuantMinima, Localizacao, Prec  
o) 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 possui clientes, independente 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.

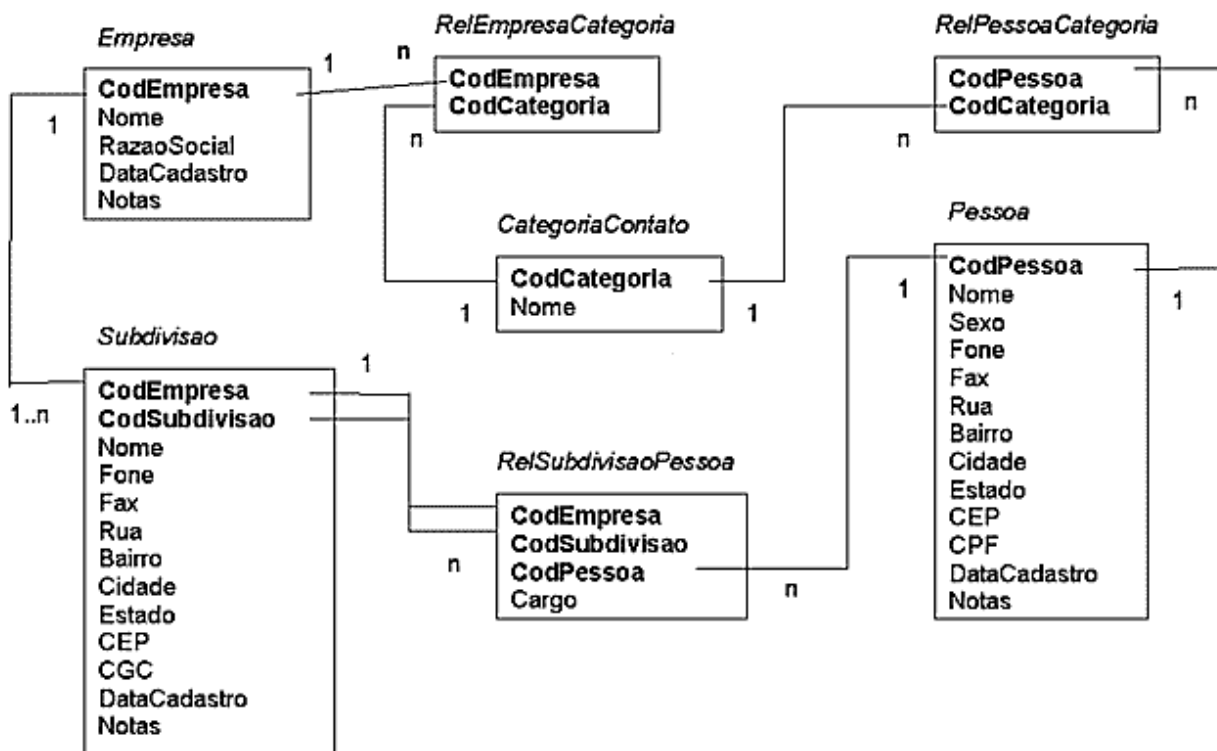
O sistema anterior da empresa, que usava arquivos DBF, tinha um cadastro de produtos. É importante que esse cadastro de produtos seja reaproveitado. Os programadores conseguiram exportá-lo para um arquivo texto, separado por tabulações. No entanto, alguns campos são diferentes do novo sistema. A gerência precisa dos seguintes relatórios:

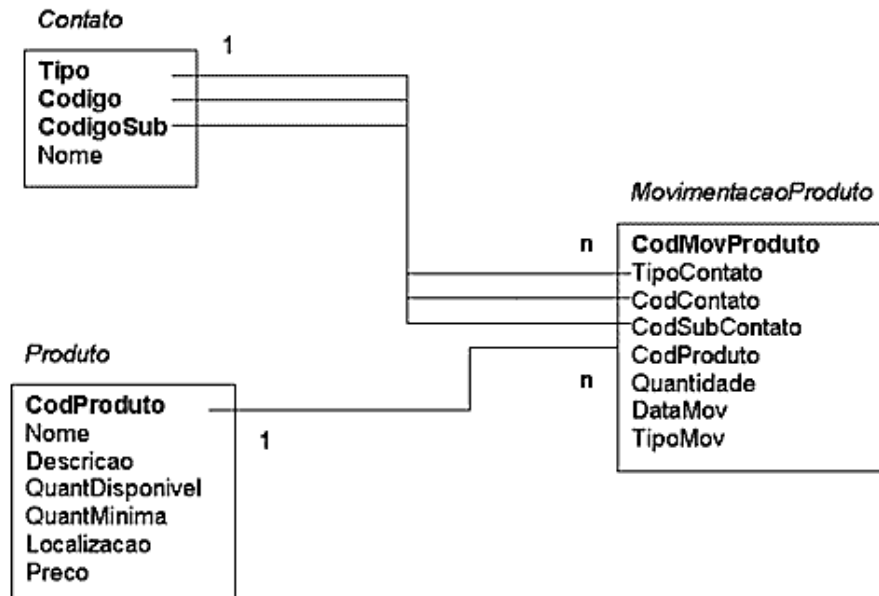
- **Quais as vendas de cada produto em determinado período?**
- **Quais as vendas e compras de cada produto num determinado período? Por exemplo:**

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





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 propriedade 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 simultaneamente, 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 versão 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 em 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 atualização. 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 simultaneamente 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.

Assistente de Atualização de Versão

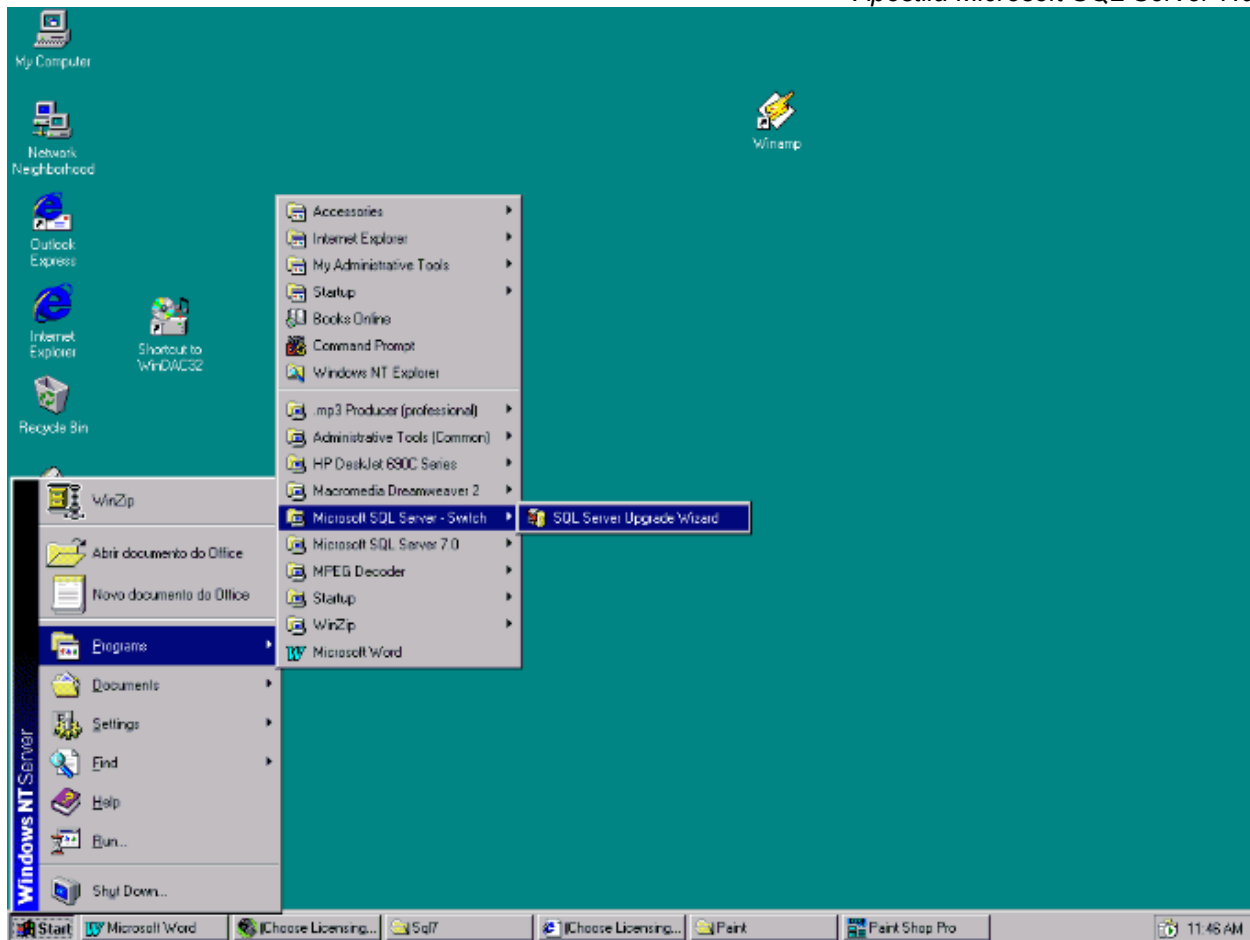
Mesmo que não se tenha o SQL Server 6.5 instalado, se durante a instalação for selecionada a opção Upgrade Tools, vai ser instalado o Assistente de atualização de versão.

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 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!

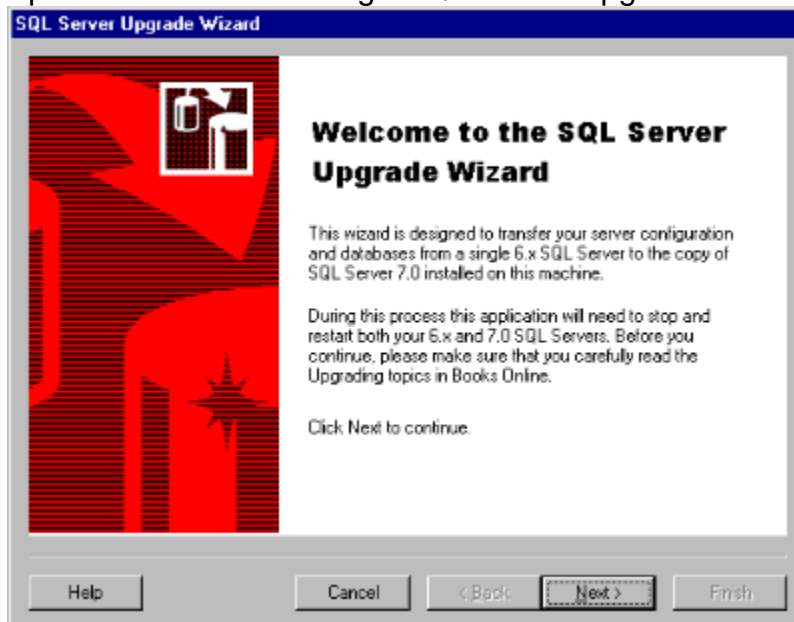
Iniciando o Assistente de Atualização de Versão



Do Menu Iniciar, selecione Microsoft SQL Server Switch e então selecione SQL Server Upgrade Wizard, como na figura abaixo:

:

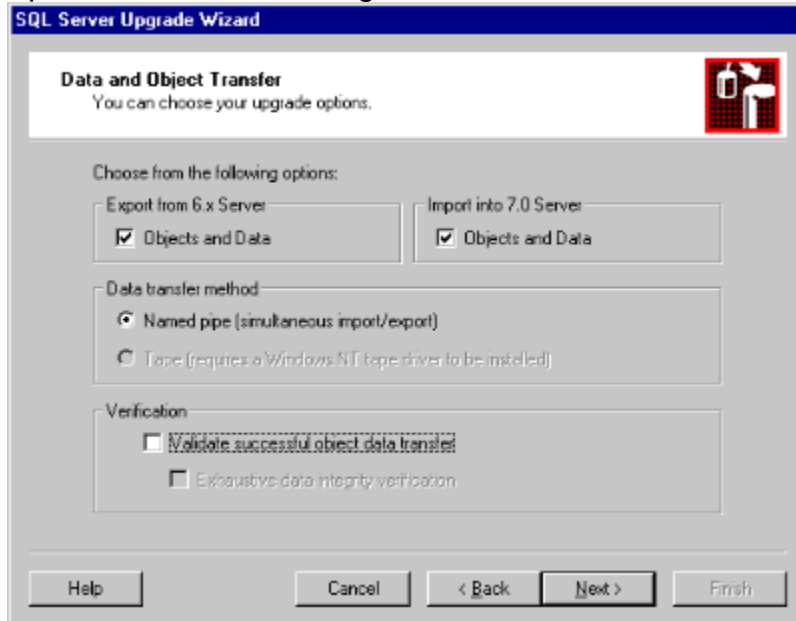
Aparece a caixa de diálogo SQL Server Upgrade Wizard, como mostrado abaixo:



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 transferência 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. Exporta 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

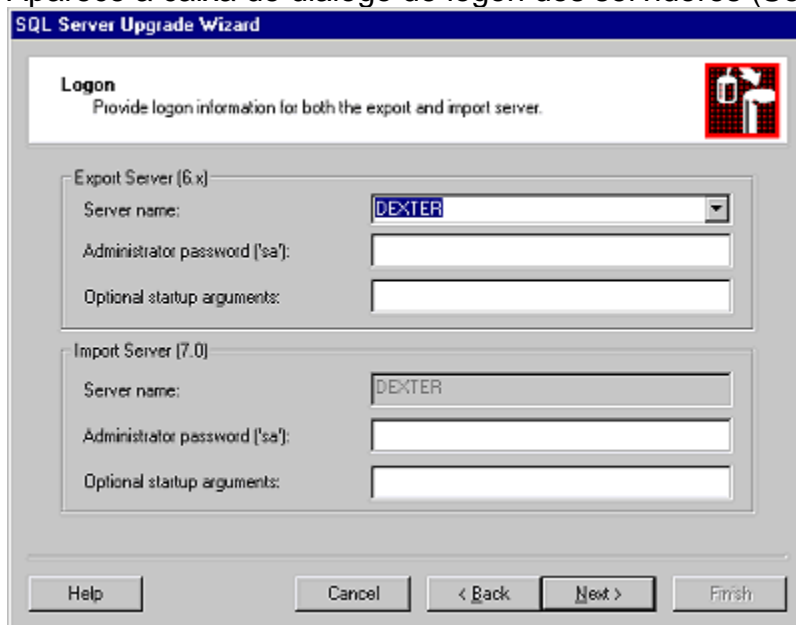
Direct Pipeline

1. Exporta os objetos 6.x
2. Fecha o SQL Server 6.x
3. Inicia 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 unidade de rede.

Logon dos servidores

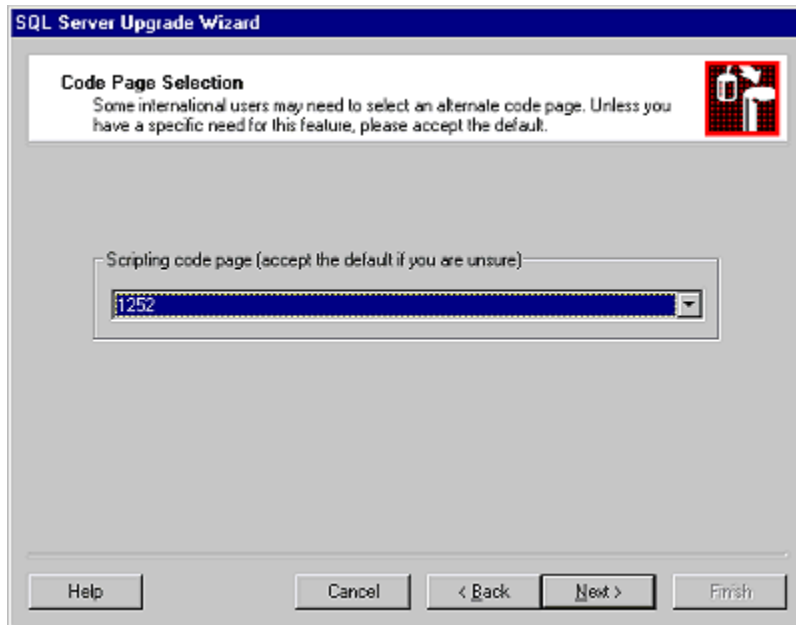
Aparece a caixa de diálogo de logon dos servidores (Servers Logon).



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.

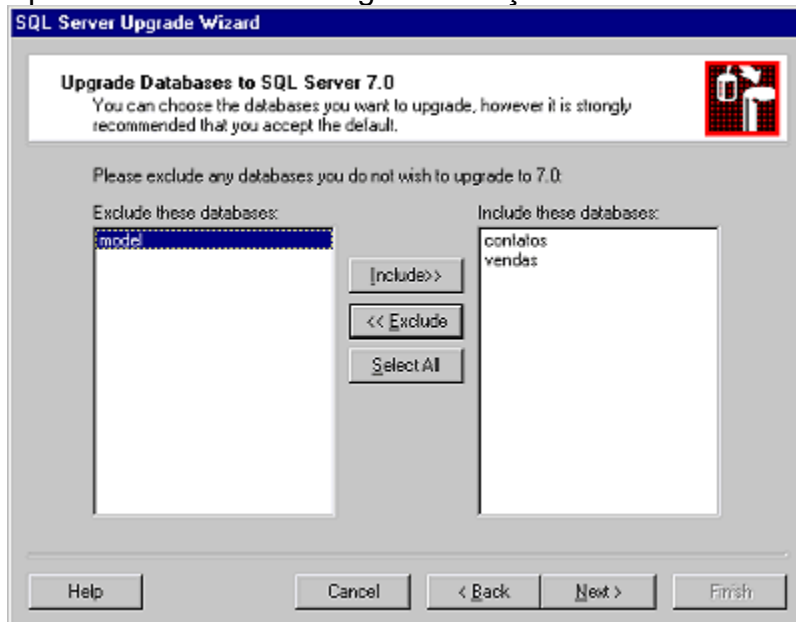
Seleção de página de código



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.

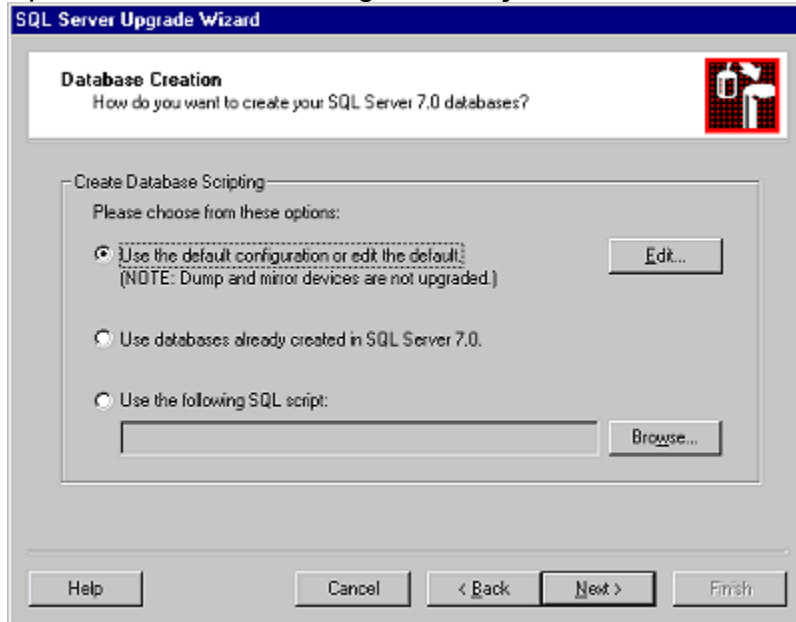


Selecione quais bancos de dados você quer atualizar colocando-os do lado "Include these databases". Os que você não quer atualizar, deixe-os do lado "Exclude these

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

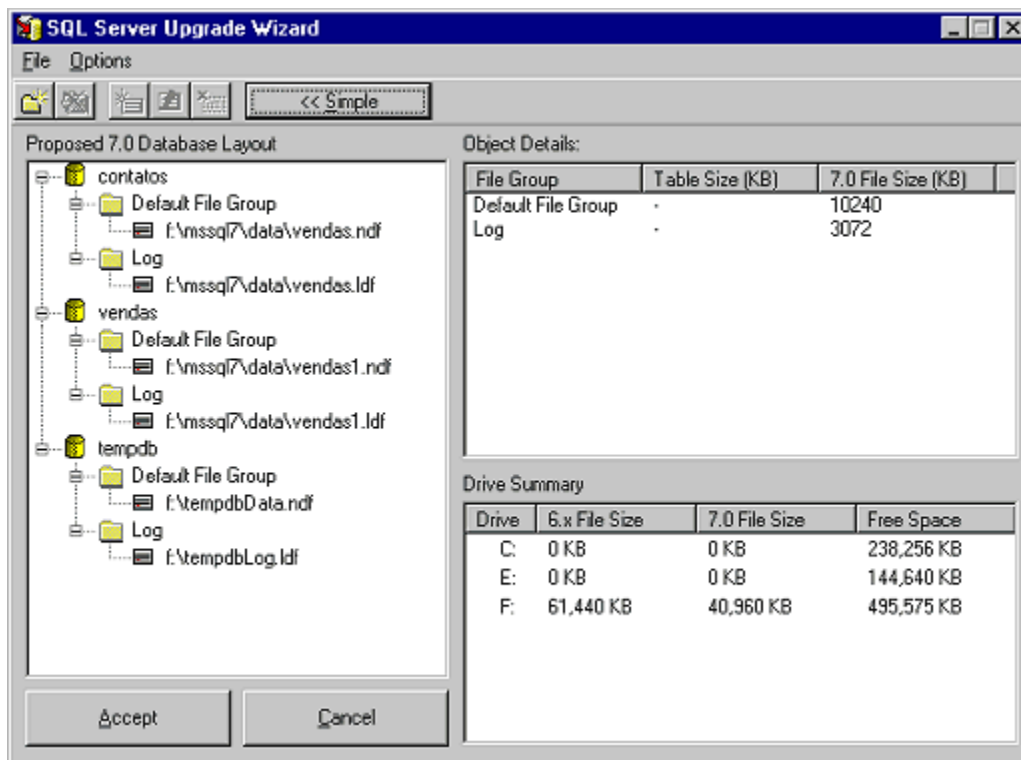
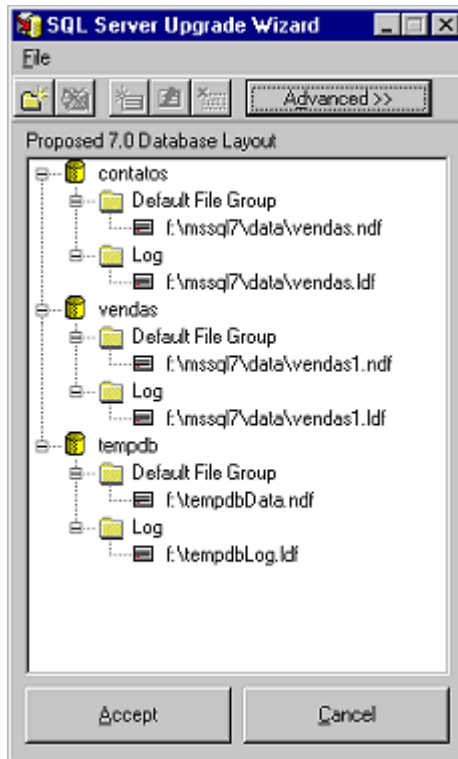
Aparece a caixa de diálogo de criação dos bancos de dados.



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".

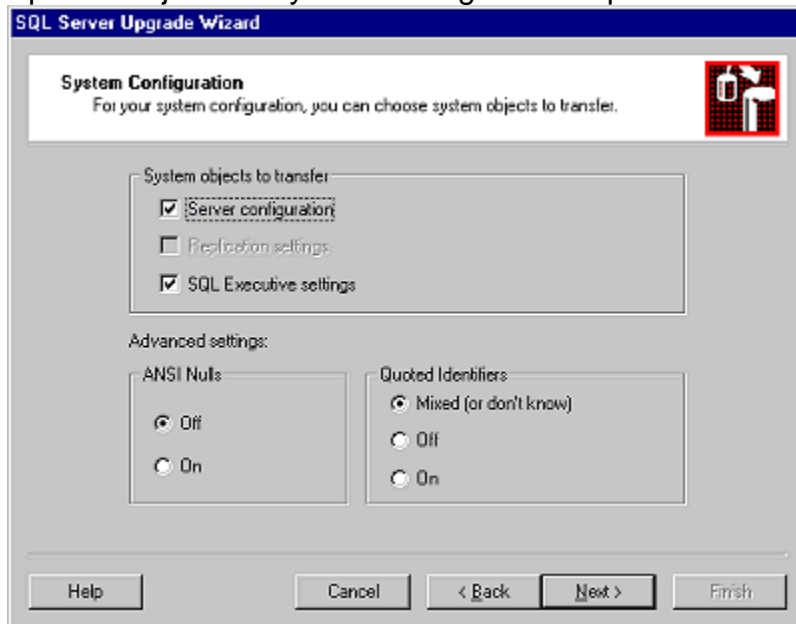


clique em Advanced e...

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

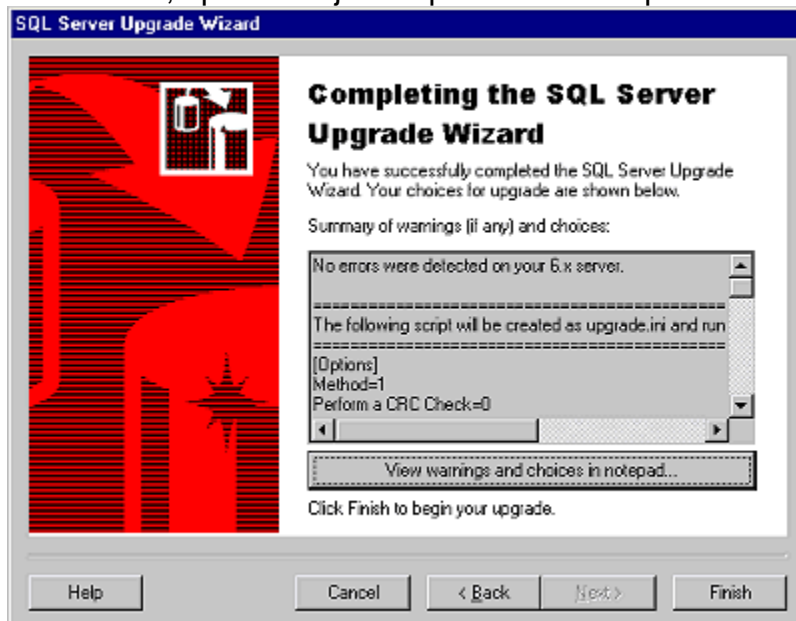
Aparece a janela "System Configuration Options".



Os valores padrão 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

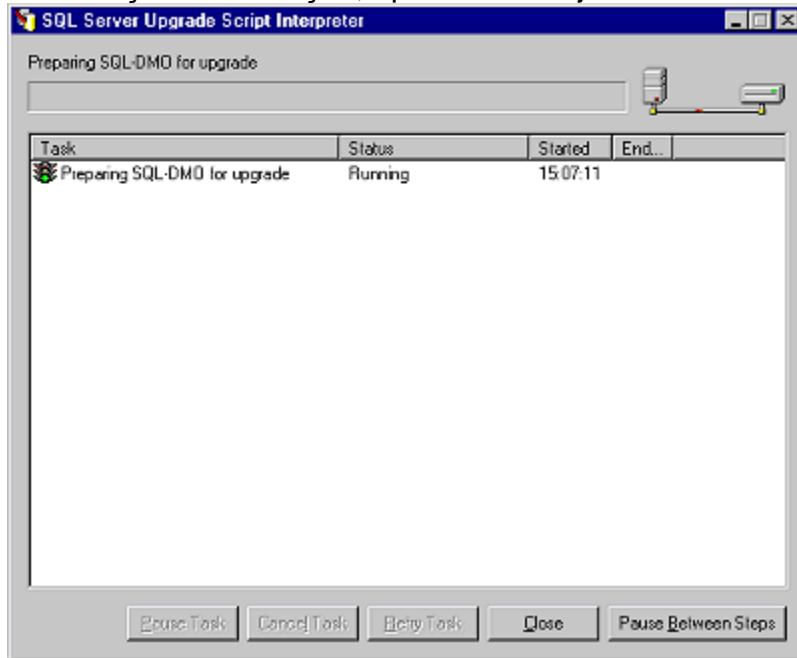
Finalmente, aparece a janela para revisar o que foi selecionado.



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 atualização. 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".