

Microsoft SQL-Server 6.5

SUMÁRIO

1. SISTEMAS DE ARQUIVOS X BANCOS RELACIONAIS.....	5
1.1 ENTIDADES, RELACIONAMENTOS E ATRIBUTOS	5
1.1.1 Entidades.....	6
1.1.2 Relacionamentos	6
1.1.3 Atributos.....	6
1.2 A LINGUAGEM SQL (<i>STRUCTURED QUERY LANGUAGE</i>)	7
2. O MICROSOFT SQL SERVER	8
2.1 BANCOS DE DADOS (<i>DATABASES</i>)	8
2.2 O BANCO DE DADOS <i>MASTER</i>	9
2.3 O BANCO DE DADOS <i>MODEL</i>	10
2.4 O BANCO DE DADOS <i>TEMPDB</i>	12
2.5 O BANCO DE DADOS <i>MSDB</i>	12
2.6 LOCALIZAÇÃO DOS BANCOS DE DADOS	13
3. USO DO PROGRAMA ISQLW.....	14
4. DECLARAÇÕES BÁSICAS DO SQL.....	21
4.1 <i>STORED PROCEDURES</i>	22
4.1.1 <i>SP_HELP</i>	23
4.1.2 <i>SP_HELPDB</i>	24
4.1.3 <i>SP_HELPTEXT</i>	24
4.1.4 <i>SP_HELPSQL</i>	25
4.2 UMA SEQÜÊNCIA BÁSICA DE TRABALHO	25
4.2.1 <i>CREATE TABLE</i>	26
4.2.2 <i>INSERT</i>	27
4.2.3 <i>SELECT</i>	28
4.2.4 <i>UPDATE</i>	29

4.2.5 DELETE.....	30
4.2.6 DROP TABLE.....	30
5. SINTAXE E EXEMPLOS DE ALGUNS COMANDOS	32
5.1 CREATE DATABASE.....	32
5.2 CREATE TABLE	32
5.3 SELECT	33
5.3.1 SELECT *	33
5.3.2 ESCOLHENDO COLUNAS	34
5.3.3 USANDO LETRAS.....	34
5.4 OPERADORES ARITIMÉTICOS	34
5.5 MANIPULAÇÃO DE DADOS NUMÉRICOS	35
5.6 MANIPULANDO CARACTERES DE DADOS	35
5.7 MANIPULANDO DADOS DE DATA E TEMPO	36
5.8 FUNÇÕES DE SISTEMA	36
5.9 CONVERSÃO DE DADOS.....	37
5.10 RECUPERAÇÃO DE DADOS	38
5.10.1 ESCOLHENDO COLUNAS	38
5.10.2 ESCOLHA DE LINHAS BASEADA EM COMPARAÇÕES	38
5.10.3 ESCOLHA DE LINHAS BASEADA EM AMPLITUDES.....	38
5.10.4 ESCOLHA DE LINHAS BASEADA EM LISTAS	39
5.10.5 ESCOLHA DE LINHAS BASEADA EM VALORES DECONHECIDOS ..	39
5.10.6 ESCOLHA DE LINHAS BASEADA EM BUSCA DE VARIOS ARGUMENTOS	39
5.10.7 ELIMINANDO DUPLICATAS	40
5.10.8 CLASSIFICANDO RESULTADOS.....	40
5.11 RECUPERAÇÃO DE DADOS - TÓPICOS AVANÇADOS	40
5.11.1 JOIN.....	40
5.11.2 Natural JOIN	40

5.11.3 <i>Eqüijoin</i>	41
5.11.4 <i>JOINS</i> com mais de duas Tabelas	41
5.11.5 <i>Auto JOINS</i>	41
5.11.6 <i>Outer JOINS</i>	41
5.12 <i>CRIANDO TRIGGERS</i>	41
5.12.1 <i>INSERT TRIGGER</i>	41
5.12.2 <i>DELETE TRIGGER</i>	42
5.12.3 <i>UPDATE TRIGGER</i>	42
5.13 <i>BULK COPY PROGRAM (BCP)</i>	42
6. ACESSO VIA INTRANET / EXTRANET / INTERNET	44
6.1 EXEMPLO PRÁTICO	47
6.2 ARQUIVOS NECESSÁRIOS E <i>SCRIPTS</i>	50
6.2.1 <i>Script</i> para o arquivo cadastro.htm	50
6.2.2 <i>Script</i> para o arquivo cadastro.idc	51
6.2.3 <i>Script</i> para o arquivo result.htx	52
6.2.4 <i>Script</i> para o arquivo todos.idc	52
6.2.5 <i>Script</i> para o arquivo cadastro.htx	53
6.2.6 <i>Script</i> para o arquivo cadpesq.idc.....	54
6.2.7 <i>Script</i> para o arquivo cadatu.htx	54
6.2.8 <i>Script</i> para o arquivo cadatu.idc	55

1. SISTEMAS DE ARQUIVOS X BANCOS RELACIONAIS

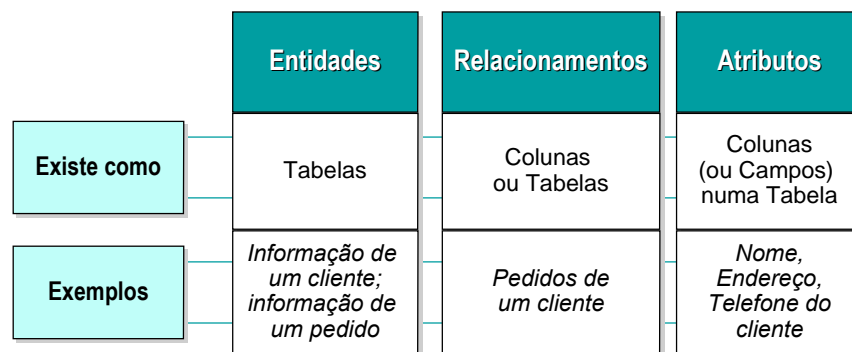
O acesso a informações em sistemas de processamento de dados que não utilizam Sistemas Gerenciadores de Bancos de Dados (SGBDs), é feito pelo acesso seqüencial a um ou mais arquivos. Cabe ao desenvolvedor criar mecanismos de recuperação da informação. Com a utilização de um SGBD, porém, o acesso fica diferente: pede-se as informações ao gerenciador de banco de dados e elas são devolvidas pelo mesmo.

O processo pode ser comparado a uma compra em uma loja de departamentos e uma compra em uma loja de autopeças, que normalmente funcionam por processo diferentes. No primeiro caso, o cliente dirige-se à loja, procura por todas as seções, encontra o produto desejado e efetua a compra. No segundo, o cliente pede ao balconista o item desejado e este entrega-o. No caso da compra em loja de departamentos, o trabalho é todo do cliente, sendo este responsável inclusive pelas especificações necessárias (fazer a escolha certa). Já na loja de autopeças, o balconista assume toda a responsabilidade pela entrega da mercadoria desejada.

1.1 ENTIDADES, RELACIONAMENTOS E ATRIBUTOS

Quanto mais organizadas estiverem as informações no Banco de Dados, mais fácil será a “conversa” com o Gerenciador de Banco de Dados.

Para isso, criou-se um modelo chamado Modelo de Entidades e Relacionamentos, do qual fazem parte três elementos:



FONTE: Microsoft

1.1.1 Entidades

Uma entidade é um objeto de interesse do qual podem ser coletadas informações. Elas são representadas por tabelas. Exemplos: tabela de clientes; tabela de pedidos de clientes.

1.1.2 Relacionamentos

As entidades podem ser relacionadas entre si pelos relacionamentos. Por exemplo: relacionamento entre a entidade de clientes e a entidade de pedidos (“clientes fazem pedidos”).

1.1.3 Atributos

Atributos são as características das entidades. São representadas pelas colunas das tabelas. Por exemplo: nome, endereço do cliente.

<i>clientes</i>				
<i>identificador</i>	<i>nome</i>	<i>endereço</i>	<i>telefone</i>	<i>.....</i>
1001	João	5554444	~~~
1002	Alberto	4687999	~~~
1003	Franciso	NULL	~~~
1004	Maria	5678900	~~~
1005	Sônia	0988855	~~~
1006	Roberto	NULL	~~~

FONTE: Microsoft

Uma das colunas de uma tabela é uma *primary key* (chave primária). Isso indica para o gerenciador de banco de dados que uma coluna (ou um conjunto de colunas) deve ter um valor único para identificar a linha inteira. **O gerenciador faz então o controle** para que não entrem duas linhas com o mesmo valor na coluna que é *primary key*.

A figura a seguir demonstra o relacionamento entre tabelas utilizando-se chaves primárias (*PK*) e estrangeiras (*FK*).

clientes						
identificador	nome
PK	NN	NN	NN	NN		
1001	João	S.....	98022	NULL	05 Jun 1992
1002	Alberto.	S.....	98022	206-555-1212	07 Ago 1992
1008	Wilson	98026	NULL	03 Mar 1993

Pedidos		
numero	cliente	produto
PK	PK,FK, NN	NN
1	1002	567
1	1001	566
2	1001	122

FONTE: Microsoft

Pedidos se relacionam aos Clientes, através do campo cliente da tabela de pedidos. Esse campo é também denominado chave estrangeira (*foreign key*). Isso garante o que é denominado integridade referencial: ou seja, não pode haver inconsistência nas linhas que estão associadas nas tabelas. Por exemplo: o **gerenciador** não permite que clientes que tenham pedidos sejam removidos da tabela clientes, nem que pedidos sejam realizados por clientes inexistentes.

1.2 A LINGUAGEM SQL (*STRUCTURED QUERY LANGUAGE*)

O SQL é uma linguagem estruturada para manipulação de dados. É padronizada para os bancos de dados relacionais, mas cada gerenciador pode possuir uma extensão própria dessa linguagem.

Como no exemplo do pedido de compra para o funcionário da loja de autopeças, cada comando no SQL é um pedido de busca ou alteração de dados para o gerenciador do banco de dados. **Quem vai executar propriamente o comando é o gerenciador.**

2. O MICROSOFT SQL SERVER

Trata-se de um Sistema Gerenciador de Bancos de Dados, Relacionais, SGBDR, que funciona unicamente sob sistema operacional *Windows NT*.

Para trabalhar com esta ferramenta a *Microsoft* fornece o ISQL, tanto em interface *DOS* quanto em interface *Windows*. Além disso, podemos nos comunicar com o banco a partir de *API's* do *Windows*, fazendo uso da camada de comunicação *DB-Library*, ou via *ODBC*. A interface com o usuário pode ser construída em *Visual Basic* ou *Visual C++*, para acesso através da *DB-Library* (que dá total controle sobre as funções do banco), ou via *VB*, *VC++*, *Visual Fox Pro*, *Access*, *Excel*, *Word*, para acesso via *ODBC*. Também podemos utilizar o acesso através de protocolo *TCP/IP* e linguagem *HTML*, caracterizando aplicações de *INTRA/INTER/EXTRANET*; o acesso ao banco propriamente dito, entre a camada de conexão a bancos de dados e o *Web Server*, será realizado via *ODBC*.

O *Microsoft SQL Server* foi originalmente baseado no *Sybase SQL Server X*, quando da versão 4.2. Na versão 6 a *Microsoft* implementou modificações visando fazer uso de características multitarefa do *Windows NT*. Atualmente está na versão 6.5, sendo aguardado para agosto/97 a versão 7, bem como uma versão *Personal*, para ambiente *Windows 95 (97)*.

2.1 BANCOS DE DADOS (DATABASES)

Uma vez instalado o *SQL Server* são criadas automaticamente quatro *databases*:

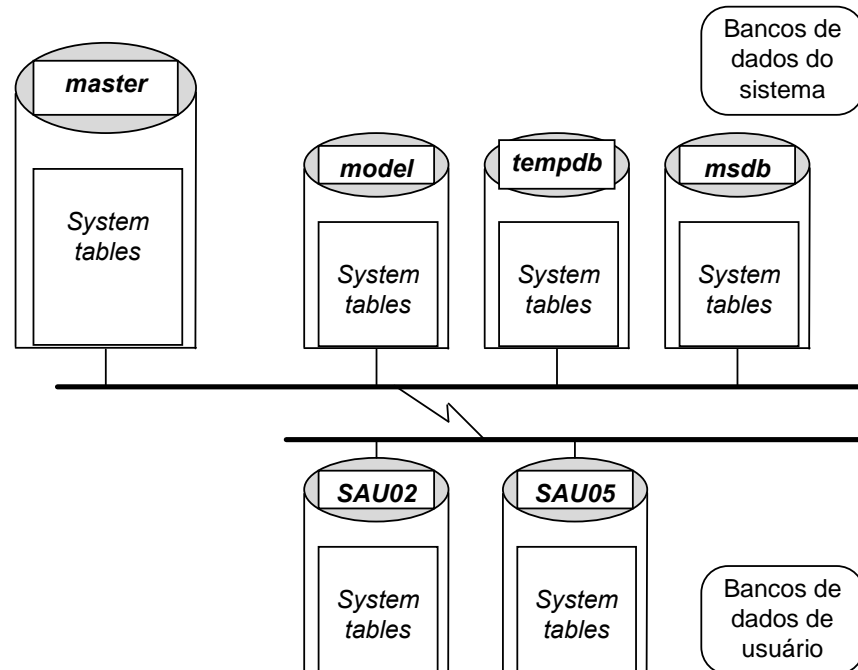
- a) *master*
- b) *model*
- c) *tempdb*
- d) *msdb*

Depois, o poderemos criar e instalar nossos próprios bancos de dados livremente, os quais serão os bancos de dados de usuário.

Embora ambos os tipos de bancos de dados (sistema e usuário) armazenem dados, o *SQL Server* utiliza os bancos de sistema para operar e gerenciar o sistema. O catálogo de sistema,

por exemplo, consiste unicamente de tabelas armazenadas no banco de dados *master*.

A figura a seguir ilustra os bancos de dados no *SQL Server*.



Vejamos a função de cada um dos bancos de sistema.

2.2 O BANCO DE DADOS *MASTER*

Controla os bancos de dados de usuários e a operação do *SQL Server*, por isso os dados armazenados em suas tabelas são críticos e deve-se sempre manter *back up* atualizado. Ocupa inicialmente cerca de 17 *Mbytes*, mantendo:

- a) contas de *login*;
- b) processos em andamento;
- c) mensagens de erro do sistema;
- d) *databases* armazenados no servidor;
- e) espaço alocado a cada *database*;
- f) *locks* ativos;
- g) *databases* disponíveis e dispositivos de *dump*;
- h) procedimentos de sistema, que são primariamente utilizados para administração.

O banco de dados *master* contém 13 tabelas de uso compartilhado com o sistema, conhecidas como Catálogo do Sistema ou Dicionário de Dados, que são:

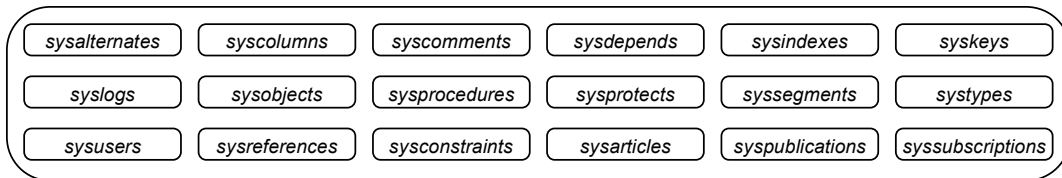
1. *syscharsets* - códigos de página que estabelecem quais caracteres estão disponíveis e sua ordem de classificação;
2. *sysconfigures* - variáveis de ambiente configuráveis;
3. *syscurconfigs* - variáveis de ambiente configuráveis;
4. *sysdatabases* - bancos existentes no servidor;
5. *sysdevices* - referência física aos dispositivos e bancos do servidor;
6. *syslanguages* - entrada para as línguas conhecidas pelo servidor;
7. *syslocks* - quais são os *locks* ativos;
8. *syslogins* - contas de usuários;
9. *sysmessages* - mensagens de erro do sistema;
10. *sysprocesses* - processos em andamento
11. *sysremotelogins* - contas de acesso remoto, para conexão entre dois servidores;
12. *syssevers* - servidores remotos;
13. *sysusages* - espaço em disco disponibilizado para cada banco de dados (relaciona-se com *sysdatabases* e *sysdevices*).

2.3 O BANCO DE DADOS *MODEL*

Fornecer um protótipo (*template*) para um novo banco de dados. Contém as tabelas de sistema que serão inseridas em cada banco de dados de usuário. As seguintes implementações podem ser realizadas neste *database*:

- a) tipos definidos pelo usuário (*user datatypes*), regras (*rules*), padrões (*defaults*), *stored procedures*;
- b) usuários que terão acesso a todos os bancos adicionados ao sistema (administradores);
- c) privilégios padrão, notadamente aos usuários *guest* (*guest accounts*);

O tamanho padrão deste banco é de 1 *Mbyte*, e sua estrutura básica pode ser vista na figura a seguir; as 18 tabelas mostradas serão sempre criadas em novos bancos de dados.



Este conjunto de 18 tabelas é conhecido como Catálogo do Banco de Dados, e suas funções são as seguintes (note que todas possuem o prefixo *sys*):

1. *sysalternates* - possui uma linha para cada usuário mapeado para um banco de dados de usuário;
2. *syscolumns* - possui uma linha para cada coluna em uma tabela ou *view*, e para cada parâmetro em uma *stored procedure*;
3. *syscomments* - possui uma ou mais linhas para cada *view*, regra (*rule*), padrão (*default*), *trigger* e *stored procedure* que contenha uma declaração de definição;
4. *sysdepends* - uma linha para cada *procedure*, *view*, ou tabela que seja referenciada por uma *procedure*, *view* ou *trigger*;
5. *sysindexes* - uma linha para cada *clustered index*, *nonclustered index*, e tabela sem índices, mais uma linha extra para cada tabela com informações de textos ou imagens;
6. *syskeys* - uma linha para cada chave estrangeira (*foreign*), primária (*primary*) ou comum (*common*);
7. *syslogs* - armazena o *transaction log*;
8. *sysobjects* - uma linha para cada tabela (*table*), visão (*view*), *stored procedure*, regra (*rule*), *trigger*, padrão (*default*), *log* e objeto temporário (somente *tempdb*);
9. *sysprocedures* - uma linha para cada visão (*view*), *stored procedure*, regra (*rule*), *trigger*, padrão (*default*);
10. *sysprotects* - mantém as informações de permissões de usuário;
11. *syssegments* - uma coluna para cada segmento;
12. *systypes* - uma linha para cada *datatype* definido pelo usuário ou fornecido pelo sistema;
13. *sysusers* - uma linha para cada usuário permitido no database;
14. *sysreferences* - uma linha para cada *constraint* de integridade referencial criada (**PK-FK**, Chave primária, chave estrangeira);
15. *sysconstraints* - informações sobre cada *constraint* criada;

As últimas três tabelas são usadas para manter informações sobre **replicação** de dados.

- 16.*sysarticles* - contém a *article information* para cada artigo criado para replicação;
- 17.*syspublications* - contém uma linha para cada publicação criada;
- 18.*syssubscriptions* - contém uma linha para cada subscrição de um *subscription server*.

2.4 O BANCO DE DADOS *TEMPDB*

Providencia um espaço de armazenamento para tabelas e outras ações temporárias ou intermediárias, tais como resultados que envolvam a cláusula *GROUP BY*, *ORDER BY*, *DISTINCT* e cursores (*CURSORS*). Possui as seguintes características:

- a) criado automaticamente no *DEVICE MASTER* (atenção, *DEVICE* e *DATABASE* são coisas diferentes);
- b) seu conteúdo é apagado quando o usuário fecha a conexão, exceto para tabelas temporárias globais;
- c) quando o banco é parado (*stopped*) seu conteúdo é apagado completamente;
- d) seu tamanho padrão é de 2 *Mbytes*.
- e) pode ser colocado em memória *RAM*.

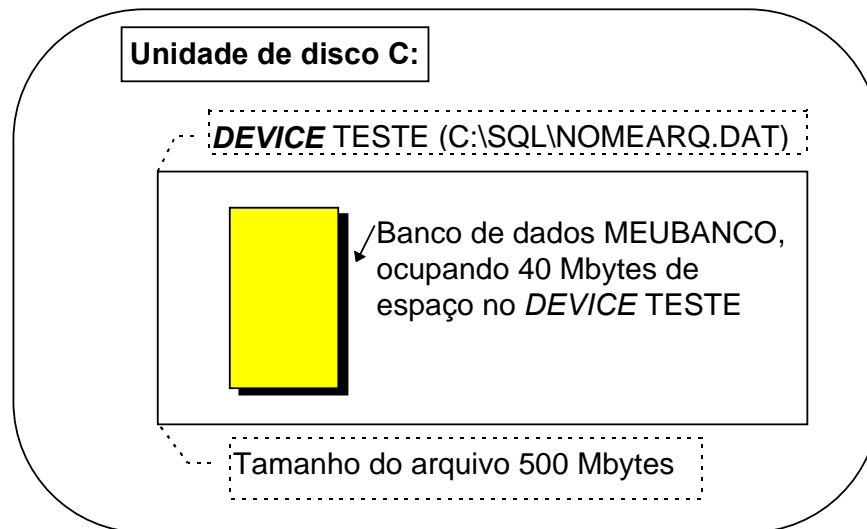
2.5 O BANCO DE DADOS *MSDB*

Providencia suporte ao **serviço** *SQL Executive Service* (o qual fornece serviços de *schedule* de tarefas, replicação, gerenciamento de alertas). Possui as seguintes tabelas de sistema:

- a) *sysalerts* - armazena informações sobre todos os alertas definidos por usuários;
- b) *sysoperators* - informações sobre os operadores;
- c) *sysnotifications* - relaciona quais operadores devem receber quais alertas;
- d) *systasks* - mantém informações sobre todas as tarefas definidas por usuários;
- e) *syshistory* - informações a respeito de quando um alerta e uma tarefa foram executados, se com sucesso ou falha, identificação do operador, data e hora da execução;
- f) *sysservermessages* - mensagens sobre as operações relacionadas ao servidor.

2.6 LOCALIZAÇÃO DOS BANCOS DE DADOS

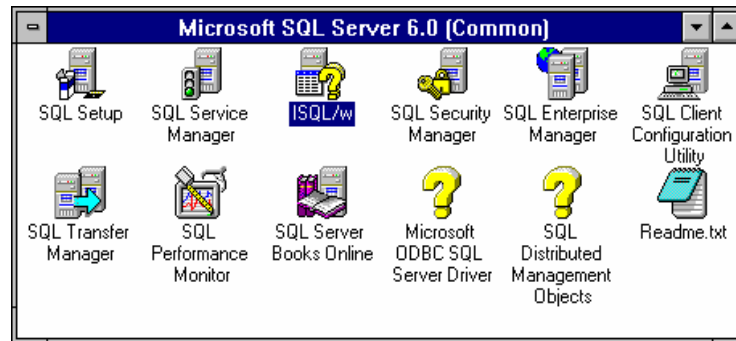
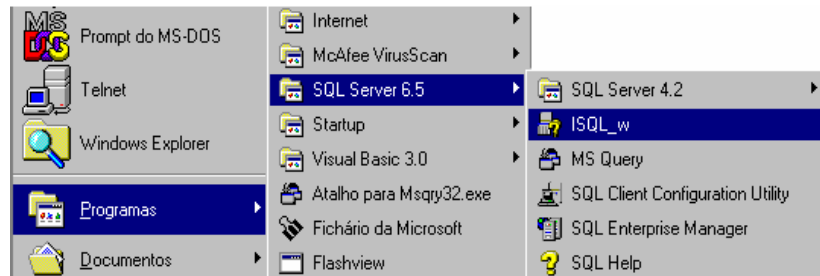
Os bancos de dados ficam armazenados em arquivos físicos que recebem o nome de *DEVICES*. Um *DEVICE* ocupa sempre a quantidade de disco que for a ele destinada, independentemente da existência ou não de bancos de dados em seu interior e independentemente da taxa de ocupação destes *databases*. Ou seja, mesmo vazio ele ocupará a porção de disco a ele destinada com seu arquivo. A figura a seguir demonstra esta característica.



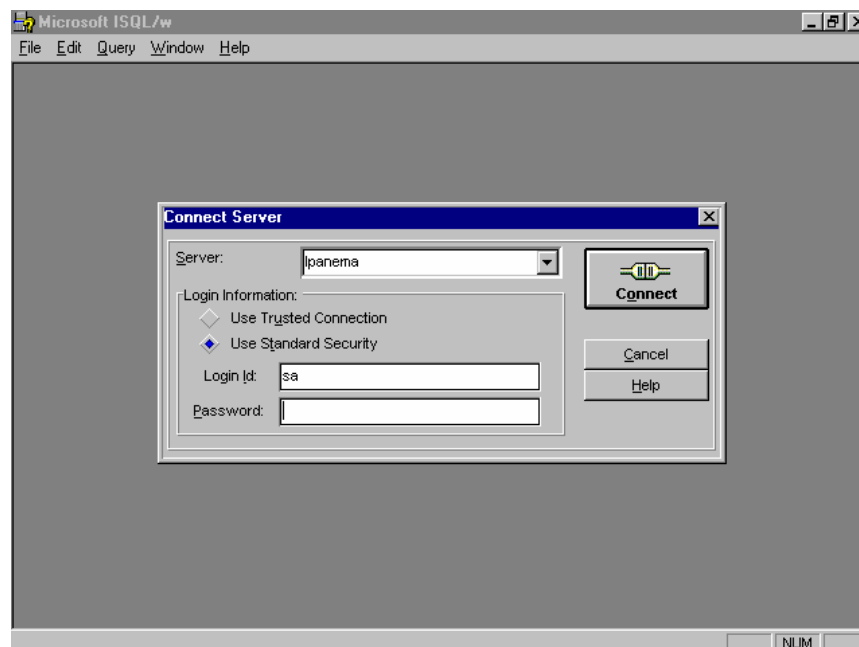
Você pode observar que existe neste exemplo um banco de dados instalado no *drive C:* (o disco rígido do equipamento), o qual contém um arquivo chamado NOMEARQ.DAT, que fisicamente ocupa 500 *Mbytes* do disco. Porém, dentro deste *DEVICE*, que recebe o nome lógico de TESTE, existe somente um banco de dados, de nome lógico MEUBANCO, o qual ocupa somente 40 *Mbytes* do espaço disponível.

3. USO DO PROGRAMA ISQLW

Localize no grupo SQL Server o ícone do ISQL/W e clique duas vezes sobre ele. Guie-se pelas figuras a seguir, conforme seu Windows seja o 95 ou o 3.11.

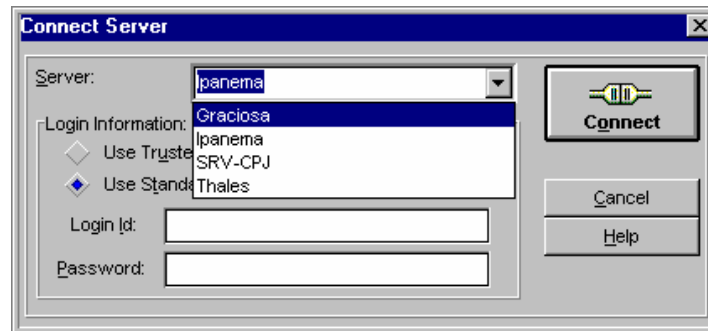


Você terá, então, uma tela como a que segue através da qual passaremos comandos ao SQL Server, após identificaremos algumas funções.


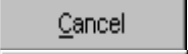
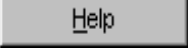


A primeira informação requerida aparece em destaque em uma pequena janela no centro da tela e refere-se à conexão com o banco de dados.

Vejamos como esta conexão será realizada.



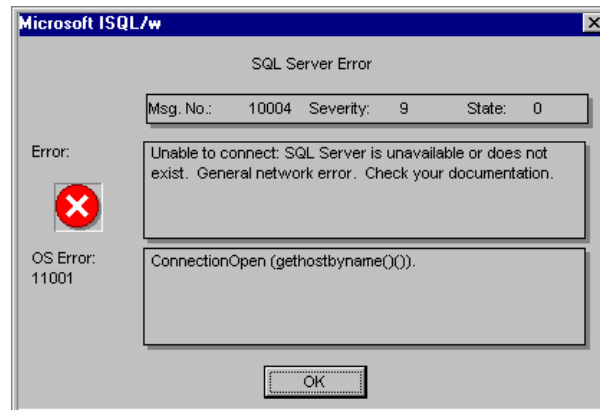
Na janela de conexão ao servidor (*Connect Server*), identificamos as seguintes funções:

Server:	<i>Combo-box</i> que mostra-nos os servidores disponíveis e reconhecidos pelo programa. Pode-se digitar o nome do servidor, caso ele não esteja presente.
Use Trusted Connection	Informa que será utilizado o <i>logon</i> padrão do Microsoft Windows.
Use Standard security	Será utilizado um objeto do MS SQL Server para controle de <i>logon</i> .
Login ID:	Nome do usuário.
Password:	Senha do usuário.
	Efetua a conexão utilizando as informações de segurança, servidor, usuário e senha fornecidas.
	Cancela a conexão, mas não fecha o ISQLW.
	Aciona o <i>Winhelp</i> com o arquivo correspondente à ajuda do ISQLW.

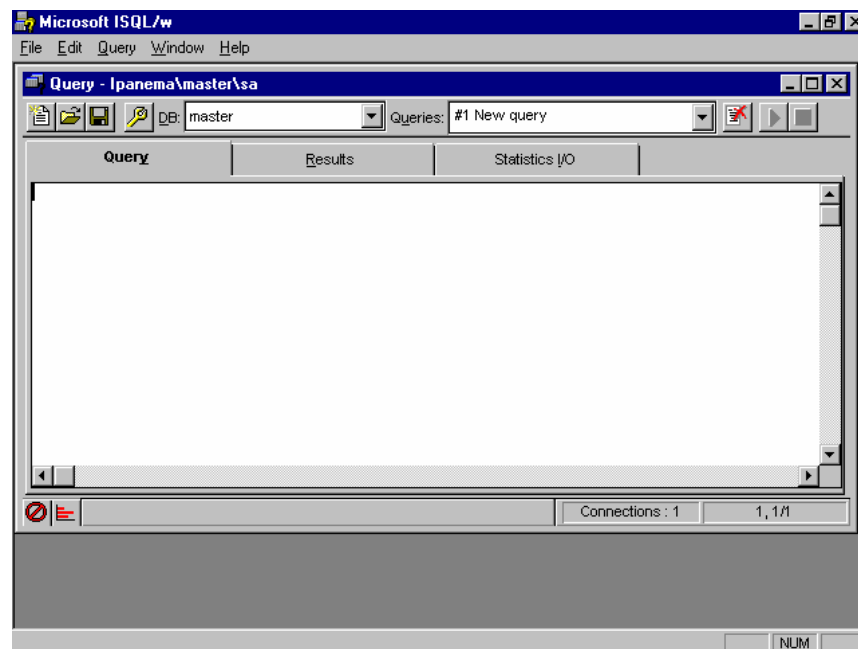
Você deverá fornecer os parâmetros adequados e iniciar sua sessão. Pergunte ao instrutor qual é o nome do servidor, qual o tipo de segurança a ser utilizado, o nome do usuário e a senha para a conexão. Saiba que se você acabou de instalar o SQL Server em sua máquina, o nome do servidor é o nome da máquina, o usuário padrão é **sa** e senha é nula (inexistente).

Caso ao iniciar a conexão surja uma tela semelhante à mostrada a seguir, experimente revisar as informações fornecidas para o logon; caso estejam corretas, verifique se o servidor está ligado e com o banco no ar; estando, verifique sua conexão de

rede. Caso estas providências não surtam efeito, contate seu suporte técnico¹.

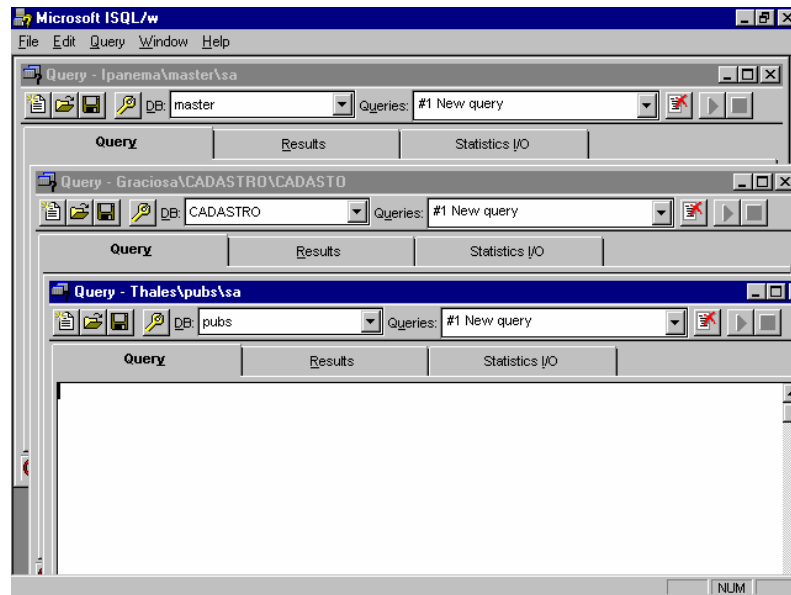


Pelo contrário, caso sua conexão tenha sido inicializada com sucesso, você terá em seu micro uma tela como a mostrada na seqüência, a qual estudaremos em seguida.

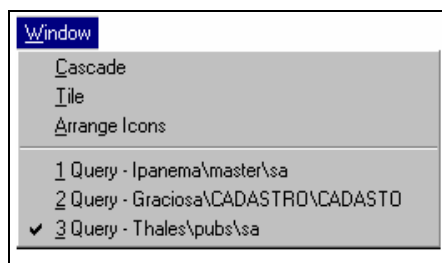


Esta tela possui uma janela principal e poderá possuir várias janelas secundárias, ou filhas. Assim, você pode utilizar o programa para gerenciar mais de uma conexão simultaneamente ou conectar-se a diferentes servidores ao mesmo tempo, como ilustrado a seguir.

¹ CCE / Microinformática - 366-2323, ramal 3116 ou simao@cce.ufpr.br



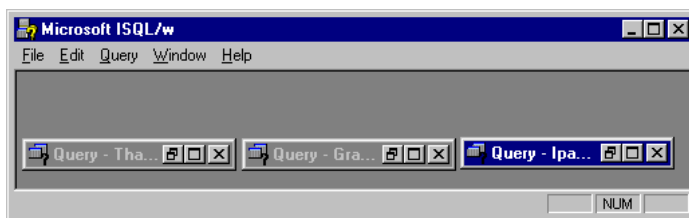
Para conectar-se a mais de um servidor, utilize o comando **File, Conect**, a partir do menu da



janela principal, informando em seguida os parâmetros necessários à conexão. Para alternar entre as diferentes janelas correspondentes às diferentes conexões, caso não estejam todas visíveis, utilize o

comando de menu Window, escolhendo a conexão desejada na lista.

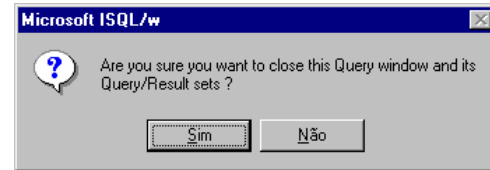
Você poderá minimizar algumas ou todas as janelas



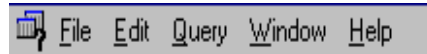
correspondentes às conexões ativas trabalhando com seus controles de estado da janela.

Windows 95	Função	Windows 3.1 / 3.11
	Minimizar	
	Maximizar	
	Restaurar	
	Fechar	

É importante ressaltar que uma vez que cada janela corresponde a uma conexão, fechada a janela, fecha-se a conexão (mas não o ISQLW).



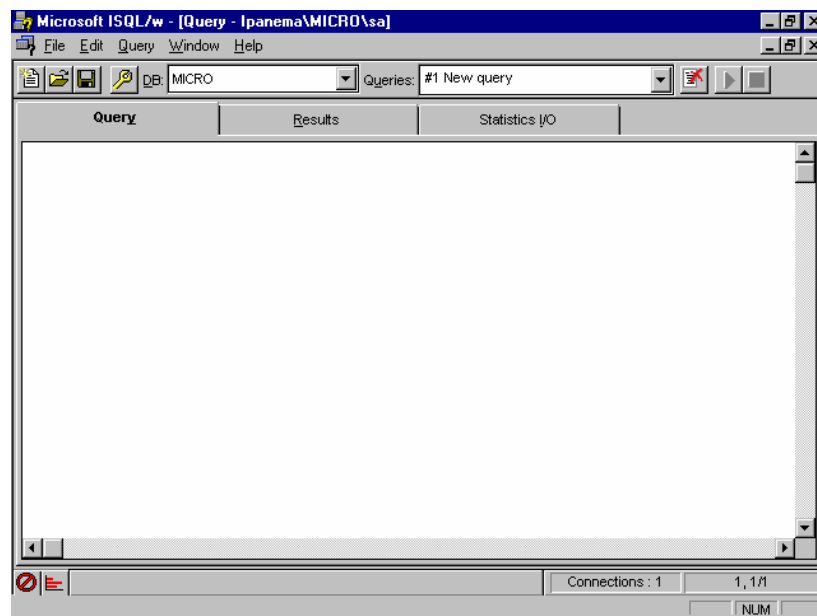
Para sejam quantas forem as janelas abertas, teremos somente uma janela principal e, por consequência, somente um menu, cujas funções são explicadas a seguir.



	Controle da janela (Restaurar, Mover, Tamanho, Minimizar, Maximizar, Fechar, Próxima).	
File	<i>Connect</i>	Abre a janela de conexão ao servidor
	<i>Disconnect</i>	Fecha uma conexão (e sua janela)
	<i>New</i>	Adiciona uma nova <i>query</i> à conexão corrente
	<i>Open</i>	Abre um arquivo contendo uma <i>query</i> ou um resultado, conforme a seleção esteja na ficha <i>Query</i> ou na <i>Results</i>
	<i>Close</i>	Termina uma <i>query</i>
	<i>Save</i>	Salva a <i>query</i> ou o resultado, conforme a seleção esteja na ficha <i>Query</i> ou na <i>Results</i>
	<i>Save as</i>	Salva com nome diferente, ou em outro diretório, ou em outro drive a <i>query</i> ou o resultado, conforme a seleção esteja na ficha <i>Query</i> ou na <i>Results</i>
	<i>Print</i>	Imprime a <i>query</i> ou o resultado, conforme a seleção esteja na ficha <i>Query</i> ou na <i>Results</i>
	<i>Print Setup</i>	Define características da impressão (papel, orientação, ...)
	<i>Configure</i>	Configura características do ISQLW, tais como o tipo de letra, características da conexão, tais como o <i>time out</i> , etc
	<i>1, 2, 3, ...</i>	Últimos arquivos utilizados
	<i>Exit</i>	Sai do ISQLW, fechando todas as conexões
Edit	<i>Undo</i>	Desfaz ações de digitação, inserção, deleção, recortar, colar, copiar (ATENÇÃO: não desfaz comandos já executados no SQL Server)
	<i>Cut</i>	Recorta o texto selecionado
	<i>Copy</i>	Copia o texto selecionado para a área de transferência
	<i>Paste</i>	Cola o conteúdo da área de transferência na posição atual do cursor ou sobre a seleção atual
	<i>Find</i>	Inicia uma busca a uma seqüência (<i>string</i>)
	<i>Repeat Last Find</i>	Repete a última busca
	<i>Replace</i>	Substitui, quando encontrada, uma seqüência por outra
Query	<i>Go to</i>	Vai para uma determinada linha. Útil para procurar informações em um conjunto de resultados, ou para procurar erros em <i>queries</i> extensas quando o SQL reporta em qual linha o erro está localizado
	<i>Execute</i>	Executa a <i>query</i> apresentada na ficha <i>Query</i>
	<i>Cancel</i>	Cancela a execução da <i>query</i> atual
	<i>Clear Window</i>	Limpa o texto da ficha atual
	<i>T_SQL Help</i>	Aciona a ajuda para o Transact SQL
	<i>Object Help</i>	Executa a <i>stored procedure sp_help</i> para o objeto selecionado
	<i>No Exec</i>	Compila a <i>query</i> mas não a executa (liga/desliga)
	<i>Statistics I/O</i>	Liga/desliga a exibição de gráficos de execução
	<i>Set Options</i>	Altera a configuração da <i>query</i>

Window	<i>Cascade</i>	Organiza as janelas abertas em cascata
	<i>Tile</i>	Organiza horizontalmente as janelas abertas
	<i>Arrange Icons</i>	Organiza os ícones de janelas minimizadas
	<i>1, 2, 3, ...</i>	Alterna para a janela ...
Help	<i>Contents</i>	Conteúdo do arquivo de ajuda do ISQLW
	<i>Transact SQL Help</i>	Ajuda específica do <i>Transact SQL</i>
	<i>Keyboard</i>	Ajuda quanto à utilização do teclado
	<i>Using Help</i>	Ajuda para utilizar a ajuda
	<i>About</i>	Informações sobre o ISQLW

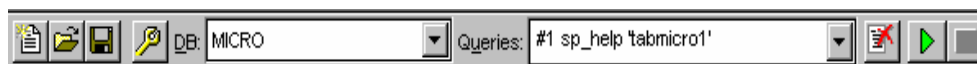
Estando com a janela correspondente ao servidor adequado aberta (pergunte ao instrutor), vamos analisá-la e iniciar com os comandos básicos.



No título da janela temos a indicação de a qual conexão ela corresponde. Isto é importante principalmente quando se possui mais de uma janela aberta.



Em seguida temos a barra de funções.



Nova consulta



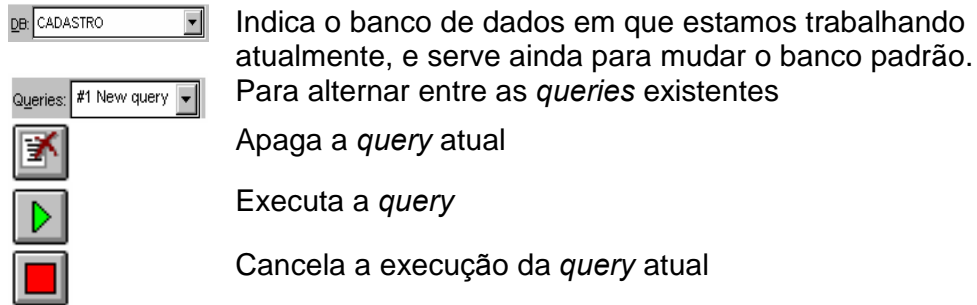
Abre uma consulta ou resultado existentes



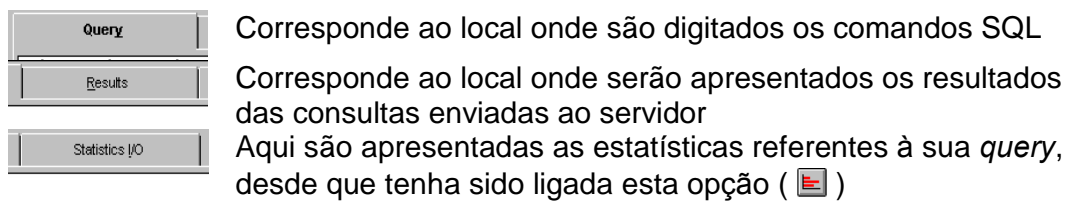
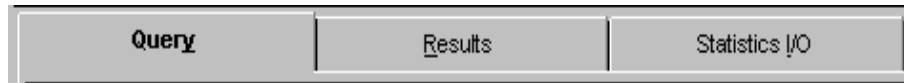
Salva uma consulta ou um resultado



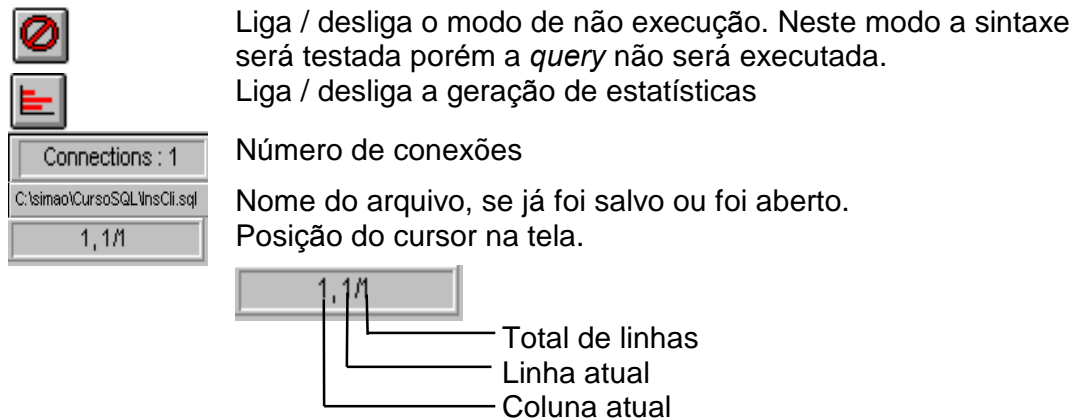
Configuração das opções de pesquisa



A seguir, uma área composta por três fichas.



Por fim, a barra de *status*.



Além dos itens mostrados anteriormente estão presentes as tradicionais barras de rolagem (*scroll bars*).

4. DECLARAÇÕES BÁSICAS DO SQL

A seguir, veremos alguns dos comandos mais utilizados na linguagem SQL, em exemplos extraídos do manual do curso da *Microsoft Implementing a SQL Server Database*.

Para iniciarmos nossos testes, iremos tomar como banco padrão o banco master; portanto, caso ele não seja o banco padrão, selecione-o.



Em seguida, na ficha *Query*, digite o seguinte comando:

```
SP_HELPDB
```

Em seguida, execute-o. Você poderá enviar um comando de execução da *query* de duas maneiras:

• Através do ícone 

• Ou pelo teclado, pressionando **Ctrl + E**

Você irá obter um resultado semelhante ao mostrado a seguir, na ficha *Results*: (evidentemente, a depender do servidor em uso, os bancos de dados serão diferentes)

Query	Results	Statistics I/O																																				
	<table border="1"> <thead> <tr> <th>name</th> <th>db_size</th> <th>owner</th> <th>dbid</th> </tr> </thead> <tbody> <tr> <td>CADASTRO</td> <td>30.00 MB</td> <td>sa</td> <td>8</td> </tr> <tr> <td>internet</td> <td>100.00 MB</td> <td>sa</td> <td>6</td> </tr> <tr> <td>master</td> <td>17.00 MB</td> <td>sa</td> <td>1</td> </tr> <tr> <td>model</td> <td>1.00 MB</td> <td>sa</td> <td>3</td> </tr> <tr> <td>msdb</td> <td>8.00 MB</td> <td>sa</td> <td>5</td> </tr> <tr> <td>pubs</td> <td>3.00 MB</td> <td>sa</td> <td>4</td> </tr> <tr> <td>sau05logico</td> <td>600.00 MB</td> <td>sa</td> <td>7</td> </tr> <tr> <td>tempdb</td> <td>94.00 MB</td> <td>sa</td> <td>2</td> </tr> </tbody> </table>	name	db_size	owner	dbid	CADASTRO	30.00 MB	sa	8	internet	100.00 MB	sa	6	master	17.00 MB	sa	1	model	1.00 MB	sa	3	msdb	8.00 MB	sa	5	pubs	3.00 MB	sa	4	sau05logico	600.00 MB	sa	7	tempdb	94.00 MB	sa	2	
name	db_size	owner	dbid																																			
CADASTRO	30.00 MB	sa	8																																			
internet	100.00 MB	sa	6																																			
master	17.00 MB	sa	1																																			
model	1.00 MB	sa	3																																			
msdb	8.00 MB	sa	5																																			
pubs	3.00 MB	sa	4																																			
sau05logico	600.00 MB	sa	7																																			
tempdb	94.00 MB	sa	2																																			

name Nome lógico do banco de dados, definido pelo criador do banco.


db_size Tamanho definido do banco de dados; corresponde ao tamanho do arquivo físico que contém o banco de dados. Este espaço será sempre ocupado na máquina, independente de o banco conter ou não informações. Funciona como um limite para o conteúdo do banco.

<code>owner</code>	Proprietário do <i>database</i> .
<code>db_id</code>	Identificação do <i>database</i> nas tabelas de sistema.
<code>created</code>	Data de criação.
<code>status</code>	Configurações e/ou opções especiais que tenham sido definidas para o <i>database</i> .

Embora tenhamos digitado o comando `sp_helpdb` usando como default o banco *master*, ele funcionaria também caso o banco *default* fosse outro. Experimente:

```
USE PUBS

SP_HELPDB
```

Note que não só você obteve o mesmo resultado, mas também seu banco default passou a ser o banco “pubs” (o que está indicado no *Combo-box DB*: ).

O comando `USE`, passado ao *SQL Server*, faz com que o banco default, ou banco de trabalho, modifique-se. Já a declaração `SP_HELPDB` corresponde a uma **stored procedure** (procedimento armazenado), que será visto a seguir.

Devemos observar que para maior clareza os comandos estão sendo digitados em letras maiúsculas. Porém, o *SQL Server* não é *case sensitive*, de maneira que podemos misturar maiúsculas e minúsculas indiferentemente.

4.1 **STORED PROCEDURES**

Stored procedures são objetos do banco de dados que contém uma série de comando SQL Padrão, que tem por objetivo facilitar e agilizar o trabalho com o banco. Podem ser de sistema ou criadas pelo usuário. Por exemplo, poderemos ter uma *stored procedure* para atualizar dados no, outra para retornar valores, outra para deletar um determinado conjunto de dados, etc.

Os procedimentos armazenados em uma *sp* são pré-compilados, de maneira que sua execução, em comparação com a execução de comandos que realizem a mesma tarefa, é mais rápida.

São usadas tanto para obter dados como para modificá-los, mas não ambos na mesma *sp*. Sua sintaxe é verificada na primeira

vez que são executadas, quando são compiladas e armazenadas em cache. Portanto, chamadas subsequentes a uma mesma *sp* serão ainda mais rápidas que a primeira.

Podem ser utilizadas em mecanismos de segurança: uma pessoa poderá possuir direitos de execução de uma *sp*, mesmo não possuindo permissões sobre as tabelas e *views* que ela referencia. Assim, por exemplo, poderíamos liberar o acesso a uma *sp* que calcula o total de salários de um determinado setor, pesquisando para isso todos os salários individuais deste setor; mas a pessoa que tivesse acesso à execução desta *sp* não teria acesso à tabela de salários propriamente dita. Como resultado, nosso usuário hipotético poderia conhecer o total de salários de cada departamento sem jamais ter contato com salários individuais.

As *stored procedures* de sistema que usaremos são: (note que todas começam com *sp_*).

SP_HELP	Fornece um relatório dos <u>objetos</u> de um <i>database</i> .
SP_HELPDB	Fornece um relatório dos <i>databases</i> existentes.
SP_HELPTEXT	Lista o texto correspondente a uma <i>stored procedure</i> e de outros objetos.
SP_HELPSQL	Exibe informações a respeito de declarações (comandos) SQL, <i>stored procedures</i> e outros tópicos.

4.1.1 SP_HELP

Quando utilizada sem parâmetros, lista todos os objetos do *database* atual:

```
SP_HELP
```

Name	Owner	Object_type
titleview	dbo	view
authors	dbo	user table
discounts	dbo	user table
employee	dbo	user table
jobs	dbo	user table
pub info	dbo	user table

Se for passado para esta *sp* o nome de uma tabela, lista todos os objetos da tabela, ou seja, exibe suas características.

```
SP_HELP authors
```

Name	Owner	Type
authors	dbo	user table
Data_located_on_segment		
default		
Column_name	Type	Length Pre
au_id	id	11
au_lname	varchar	40
au_fname	varchar	20
phone	char	12
address	varchar	40
city	varchar	20

4.1.2 SP_HELPDB

Fornece uma lista dos databases.

SP_HELPDB

name	db_size	owner	dbid	c
CADASTRO	30.00 MB	sa	8	J
internet	100.00 MB	sa	6	J
master	17.00 MB	sa	1	A
model	1.00 MB	sa	3	A
msdb	8.00 MB	sa	5	J
pubs	3.00 MB	sa	4	A
sau05logico	600.00 MB	sa	7	J
tempdb	94.00 MB	sa	2	J

4.1.3 SP_HELPTEXT

Lista o texto correspondente a uma *sp* e de outros objetos.

SP_HELPTEXT sp_help

Name	Owner	Type
sp_help	dbo	stored proc
Data_located_on_segment		
not applicable		
Parameter_name	Type	Length Pre
@objname	varchar	92 92

Note que, como a *stored procedure* SP_HELP está armazenada no *database master*, será necessário alternar para este banco antes de iniciar o comando, caso contrário será visualizada a mensagem de erro a seguir, indicando que o objeto não foi encontrado no *database* em uso.

```
Msg 15009, Level 16, State 1
The object 'sp_help' does not exist in database 'pubs'.
```


4.1.4 SP_HELPSQL

Exibe informações a respeito de declarações (comandos) SQL, *stored procedures* e outros tópicos.

Caso não seja passado um parâmetro, a *sp* SP_HELPSQL exibirá uma janela com informações:

```
SP_HELPSQL
```

```
sp_helpsql supplies help on Transact-SQL statements, server-supplied sto
  procedures, and the following special topics:
```

Comments	Expression	Variables
Control of Flow	Functions	Wildcards
Cursors	Operators	
Datatypes	Transactions	

```
Syntax: sp_helpsql ['topic']
```

For parameter options and syntax restrictions, see the Books On-line.

Para passar como parâmetro o comando sobre o qual se necessita de ajuda, devermos passá-lo entre aspas, pois caso contrário surgirá uma mensagem de erro. As aspas poderão ser simples ou duplas, desde que ambas (início e fim) sejam do mesmo tipo. Para maior clareza, e com fins de padronização, prefira aspas simples.

```
SP_HELPSQL 'select'
```

```
Transact-SQL Syntax Help
```

```
-----
SELECT Statement
Retrieves rows from the database.
```

```
SELECT [ALL | DISTINCT] <select_list>
      INTO [<new_table_name>]
      [FROM <table_name> [, <table_name2> [... , <table_name16>]]]
      [WHERE <clause>]
      [GROUP BY <clause>]
      [HAVING <clause>]
      [ORDER BY <clause>]
      [COMPUTE <clause>]
      [FOR BROWSE]
```

4.2 UMA SEQÜÊNCIA BÁSICA DE TRABALHO

Vamos providenciar a criação de uma tabela, na qual iremos inserir algumas linhas, para depois selecioná-las e alterá-las, fechando assim um ciclo de comandos SQL básicos, os quais

serão posteriormente analisados. Ao final, apagaremos nossa tabela de teste.

4.2.1 CREATE TABLE

Para criarmos uma tabela, deveremos utilizar a declaração `CREATE TABLE`, unindo a ela o nome que será atribuído ao objeto e suas características. Para verificar a sintaxe completa, use a declaração vista anteriormente `SP_HELP SQL 'CREATE TABLE'`.

```

USE master ( DB: master )

CREATE TABLE cliente

(
    cliente numeric (8,0) not null PRIMARY KEY,
    nome varchar (60) null,
    telefone varchar (20) null
)

+++++

CREATE TABLE pedidos

(
    numero int NOT NULL ,
    cliente numeric(8, 0) NOT NULL ,
    telefone int NOT NULL ,
    PRIMARY KEY
        (
            cliente,
            numero
        ),
    FOREIGN KEY
        (
            cliente
        )

```

```

REFERENCES cliente
(
    cliente
)
)

```

4.2.2 **INSERT**

Para inserirmos dados em uma tabela, devemos informar qual é a tabela, quais os campos que estamos inserindo e quais são seus valores.

```

INSERT cliente (cliente, nome, telefone)
values (1001, 'João', '445-0988')

```

```

INSERT cliente (cliente, nome, telefone)
values (1002, 'Alberto', '465-9887')

```

```

INSERT cliente (cliente, nome, telefone)
values (1003, 'Maria', '789-9877')

```

```

INSERT cliente (cliente, nome, telefone)
values (1004, 'Sônia', null)

```

A ordem dos campos pode ser diferente da ordem que estes possuem na tabela:

```

INSERT cliente (nome, cliente, telefone)
values ('Carlos', 1005, null)

```

Caso existam valores para todos os campos, podemos omitir seus nomes.

```

INSERT cliente
values (1006, 'Viu só?', '999-0000')

```

4.2.3 SELECT

Através do comando `select`, recuperamos os dados existentes no banco, de acordo com os critérios desejados:

```
SELECT nome, telefone FROM cliente
```

nome	telefone
João	445-0988
Alberto	465-9887
Maria	789-9877
Sônia	(null)

(4 row(s) affected)

Podemos recuperar todas as colunas de uma tabela utilizando o caracter curinga `*`.

```
SELECT * FROM cliente
```

cliente	nome	telefone
1001	João	445-0988
1002	Alberto	465-9887
1003	Maria	789-9877
1004	Sônia	(null)
1005	Carlos	(null)
1006	Viu só?	999-0000

(6 row(s) affected)

A utilização da cláusula `WHERE` faz com que o uso do comando `SELECT` seja dos mais freqüentes no dia a dia, pois através dela poderemos especificar condições de busca, as quais determinarão a quantidade de informações retornadas pelo servidor, ou, muitas vezes, trarão exatamente o que precisamos. Esta última característica, de obtermos exata e somente aquilo que necessitamos é que faz a grande diferença entre um servidor de arquivos, que envia pela rede o arquivo todo, e um gerenciador de bancos de dados, que envia somente o suficiente.

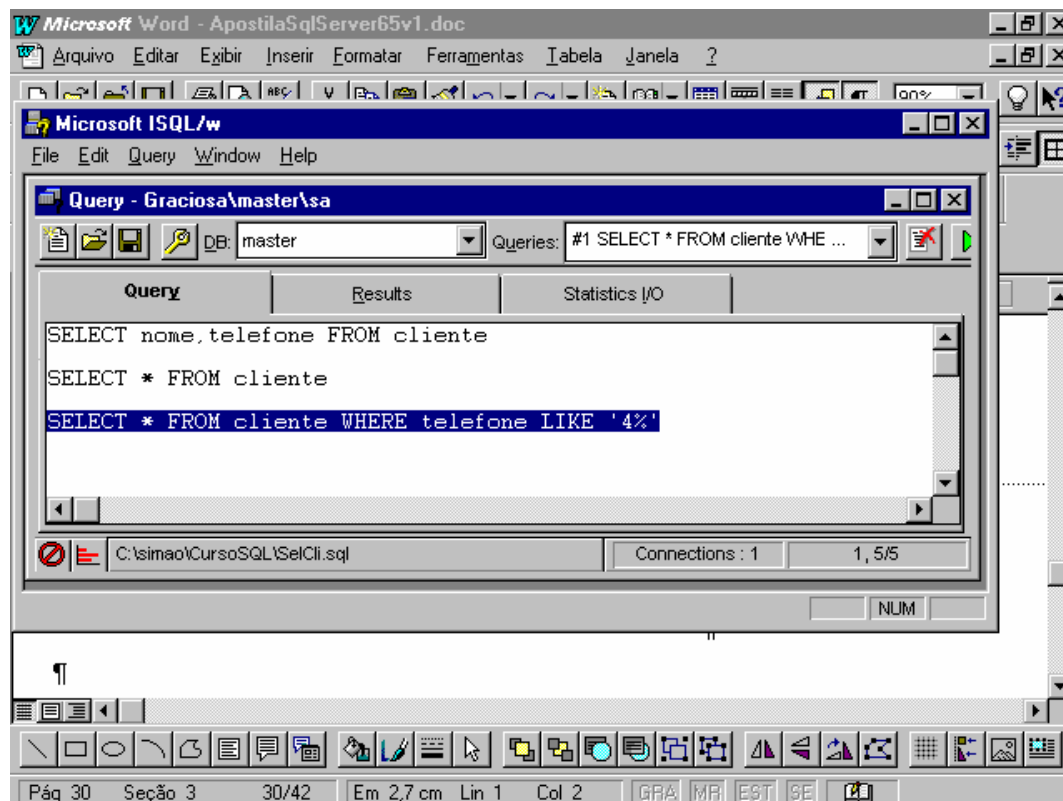
```
SELECT * FROM cliente WHERE telefone LIKE '4%'
```

cliente	nome
1001	João
1002	Alberto

(2 row(s) affected)

Observação

Caso exista mais de uma sentença em sua janela de queries, e você deseje executar apenas uma, selecione a sentença que você deseja executar, antes de comandar sua execução. O que não estiver selecionado será ignorado.



4.2.4 UPDATE

Utilizado para modificar dados já cadastrados. Pode ser usado para atualizar todas as linhas ou para atualizar linhas que correspondam a determinados critérios.

```
UPDATE cliente SET telefone = '000-1111'
```

cliente	nome	telefone
1001	João	000-1111
1002	Alberto	000-1111
1003	Maria	000-1111
1004	Sônia	000-1111
1005	Carlos	000-1111
1006	Viu só?	000-1111

(6 row(s) affected)

```
UPDATE cliente
```

```
SET telefone = '111-0000'
```

```
WHERE cliente = 1004
```

cliente	nome	telefone
1001	João	000-1111
1002	Alberto	000-1111
1003	Maria	000-1111
1004	Sônia	111-0000
1005	Carlos	000-1111
1006	Viu só?	000-1111

```
(6 row(s) affected)
```

4.2.5 DELETE

Para apagar linhas de uma tabela devemos especificar critérios, através da cláusula `WHERE`; caso contrário todas as linhas da tabela serão apagadas. A tabela, porém, não será eliminada. Continuará existindo, porém vazia.

```
DELETE cliente
```

```
WHERE cliente = 1006
```

cliente	nome	telefone
1001	João	000-1111
1002	Alberto	000-1111
1003	Maria	000-1111
1004	Sônia	111-0000
1005	Carlos	000-1111

```
(5 row(s) affected)
```

```
DELETE cliente
```

cliente	nome	telefone
---------	------	----------

```
(0 row(s) affected)
```

4.2.6 DROP TABLE

Este procedimento irá remover a tabela completamente, não existindo nenhum procedimento de *recovery*. Portanto, assegure-se de que a tabela em questão realmente não é mais necessária, ou, pelo menos, faça um *back up* do banco antes.

Ao apagar uma tabela, saiba que os relacionamentos por ventura com ela existente impedirão sua deleção. Por isso, você deverá começar a apagar as tabelas desde as “filhas”.

```
DROP TABLE cliente
```

Msg 3726, Level 16, State 1

Could not drop object 'cliente'.

It is being referenced by a foreign key constraint.

Assim, para apagar a tabela cliente, antes será necessário apagar a tabela pedidos.

```
DROP TABLE pedidos
```

```
DROP TABLE cliente
```

5. SINTAXE E EXEMPLOS DE ALGUNS COMANDOS

Sintaxe e exemplos a seguir foram retirados do *Help* do *SQL Server*, que possui informações bem mais completas que o resumo aqui apresentado.

O banco a que se referem os exemplos é o `pubs`, instalado juntamente com o *SQL Server*. Caso por qualquer motivo o banco de exemplos `pubs` não esteja presente em sua instalação, procure os *scripts* de instalação no subdiretório *install* e execute-os. Assim você poderá testar os exemplos.

5.1 CREATE DATABASE

Sintaxe:

```
CREATE DATABASE database_name
[ON {DEFAULT | database_device} [= size]
[, database_device [= size]]...
[LOG ON database_device [= size]
[, database_device [= size]]...
[FOR LOAD]
```

OBS.: Tamanhos em megabytes

Exemplos:

1. CREATE DATABASE pubs (o tamanho default é 2 Mb)
2. CREATE DATABASE newpubs
ON default = 256
3. CREATE DATABASE newdb
ON default = 50, newdata = 25
4. CREATE DATABASE library
ON library_dev1 = 10
LOG ON librlog_dev2 = 4

5.2 CREATE TABLE

Tipos de dados	Tipos de dados supridos pelo sistema
Binary	binary[(n)], varbinary[(n)]
Character	char[(n)], varchar[(n)]
Date and time	datetime, smalldatetime
Exact numeric	decimal[(p,s)]
Approximate numeric	float[(n)], real
Integer	int, smallint, tinyint
Monetary	money, smallmoney

Special	bit, timestamp, user-defined datatypes
Text and imagem	text, imagem
Synonyms	binary, varying for varbinary, character for char, character, varying for varchar, dec for decimal, integer for int, double precision for float

Sintaxe:

```
CREATE TABLE [database.[owner].]table_name
(
  { col_name column_properties[constraint[constraint[...constraint]]]
  | [[.] constraint]}
  [[.] {next_col_name|next_constraint}...]
)
[ON segment_name]
```

Exemplos:

<i>Nome da coluna</i>	<i>Tipo de Dados</i>	<i>Null ou não Null</i>
CREATE TABLE member		
(
member_no	member_no	NOT NULL,
lastname	shortstring	NOT NULL,
firstname	shortstring,	
middleinitial	letter	NULL
photograph	image	NULL
)		

5.3 **SELECT**

Sintaxe:

```
SELECT[ALL|DISTINCT] select_list
  [INTO[ new_table_name ]]
[FROM{table_name|view_name}{optimizer_hints}
  [[,{table_name2|view_name2}{optimizer_hints}]
  [...,{table_name16|view_name16}[(optimizer_hints)]]]
[WHERE clause]
[GROUP BY clause]
[HAVING clause]
[ORDER BY clause]
[COMPUTE clause]
[FOR BROWSE]
```

5.3.1 **SELECT ***

Sintaxe: SELECT *

```
FROM table_name
```

5.3.2 ESCOLHENDO COLUNAS

```
SELECT column_name [, column_name ...]
FROM table_name
```

```
SELECT au_id, au_fname, au_lname
FROM authors
```

5.3.3 USANDO LETRAS

```
SELECT column_name | 'string literal' [, column_name 'string_literal' ...]
FROM table_name
```

```
SELECT au_fname, au_name, 'Identification number:',
au_id
FROM authors
```

```
SELECT column_heading = column_name [, column_name ...]
FROM table_name
```

ou

```
SELECT column_name column_heading [, column_name ...]
FROM table_name
```

```
SELECT FIRST = au_fname, LAST = au_lname,
IDENTIFICATIO# =
'Identification number:', Author_ID = au_id
FROM authors
```

5.4 OPERADORES ARITIMÉTICOS

Operação	tipos de dados que podem usar esta operação
+	int, smallint, tinyint, numeric, decimal, float, real, money e smallmoney
-	int, smallint, tinyint, numeric, decimal, float, real, money e smallmoney
/	int, smallint, tinyint, numeric, decimal, float, real, money e smallmoney
*	int, smallint, tinyint, numeric, decimal, float, real, money e smallmoney
%	int, smallint e tinyint

Sintaxe

```
{ constant | column_name | function | (subquery) }
[{ arithmetic_operator | bitwise_operator | string_operator }
{ constant | column_name | function | (subquery) } ...]
```

```
SELECT price, (price * 1.1), title
FROM titles
```

5.5 MANIPULAÇÃO DE DADOS NUMÉRICOS

Função	Parâmetros
ABS	(numeric_expr)
ACOS,ASIN,ATAN,ATN2	(float_expr)
COS,SIN,COT,TAN	(float_expr)
CEILING	(numeric_expr)
DEGREES	(numeric_expr)
EXP	(float_expr)
FLOOR	(numeric_expr)
LOG	(float_expr)
LOG10	(float_expr)
PI	()
POWER	(numeric_expr,y)
RADIANS	(numeric_expr)
RAND	([seed])
ROUND	(numeric_expr,length)
SIGN	(numeric_expr)
SQRT	(float_expr)

```
SELECT title_id,
       ROUND(price*royalty/100,0)
FROM titles
```

5.6 MANIPULANDO CARACTERES DE DADOS

Função	Parâmetros
+	(expression expression)
ASCII	(char_expr)
CHAR	(integer_expr)
CHARINDEX	('pattern', expression)
DIFFERENCE	(char_expr1,char_expr2)
LOWER	(char_expr)
LTRIM	(char_expr)
PATINDEX	('%patern%', expression)
REPLICATE	(char_expr, integer_expr)
REVERSE	(char_expr)
RIGHT	(char_expr,integer_expr)
RTRIM	(char_expr)
SOUNDEX	(char_expr)
SPACE	(integer_expr)

Função	Parâmetros
STR	(float_expr[,length[,decimal]])
STUFF	(char_expr1, start, length, char_expr2)
SUBSTRING	(expression, start, length)
UPPER	(char_expr)

```
SELECT au_lname + ', ' +
       Substring (au_fname,1,1) + '.',
       au_id
FROM authors
```

5.7 MANIPULANDO DADOS DE DATA E TEMPO

FUNÇÃO	PARAMETROS
DATEADD	(datepart, number, date)
DATEDIFF	(datepart, date1, date2)
DATENAME	(datepart, date)
DATEPART	(datepart, date)
GETDATE	()

Tipos de data	Abreviações	Valores aceitos
year	yy	1752-9999
quarter	qq	1-4
mont	mm	1-12
day of year	dy	1-366
day	dd	1-31
week	wk	0-51
weekday	dw	1-7 (1 é domingo)
hour	hh	0-23
minute	mi	0-59
second	ss	0-59
millisecond	ms	0-999

```
SELECT
       DATEDIFF (MONTH, pubdate, GETDATE())
FROM Titles
```

5.8 FUNÇÕES DE SISTEMA

FUNÇÃO	PARÂMETROS
COALESCE	(expression1,expression2,...expressionN)
COL_NAME	('table_id', column_id)
COL_LENGTH	('table_name', 'column_name')
DATALenght	('expression')
DB_ID	(['databasename'])
DB_NAME	([database_id])

FUNÇÃO	PARÂMETROS
GETANSINULL	(['databasename'])
HOST_ID	()
HOST_NAME	()
IDENT_INCR	('table_name')
IDENT_SEED	('table_name')
INDEX_COL	('table_name', index_id, key_id')
ISNULL	(expression, value)
NULLIF	(expression1,expression2)
OBJECT_ID	('object_name')
OBJECT_NAME	(object_id)
STATS_DATE	(table_id,index_id)
SUSER_ID	(['server_user_id'])
SUSER_NAME	([server_user_id])
USER_ID	(['username'])
USER_NAME	([user_id])

```
SELECT length = DATALENGTH(pub_name), pub_name
FROM publishers
```

```
Resultado: length      pub_name
           14         New Moon Books
           16         Binnet & Hardley
           20         Algodata Infosystems
           21         Five Lakes Publishing
(4 row(s) affected)
```

5.9 CONVERSÃO DE DADOS

CONVERT(datatype[(length)],expression[,style])

COM SEC.	SEC.	STANDARD	FORMATO DE SAIDA DOS DADOS
1	101	USA	mm/dd/yy
2	102	ANSI	yy.mm.dd
3	103	britânico	dd/mm/yy
10	110	USA	mm-dd-yy
12	112	ISO	yymmdd

```
SELECT 'Title Code' = pub_id +
UPPER(SUBSTRING(type,1,3)) +
```

```
SUBSTRING(CONVERT(CHAR(4),DATEPART(YY,pubdate)),3,3)
FROM titles
```

```
Resultado: Title Code
           1389BUS91
           0736BUS91
```

```

1389BUS91
.
.
.
(18 row(s) affected)

```

5.10 RECUPERAÇÃO DE DADOS

Existem muitas variações e usos para o comando SELECT. Vejamos algumas.

5.10.1 ESCOLHENDO COLUNAS

```

SELECT select_list
      FROM table_list
      WHERE search_conditions

```

Condições de pesquisa incluídas:

- Operadores de comparação (=,>,<,<=,>=,<>,!<,>!, e !>)
- Amplitude (BETWEEN and NOT BETWEEN)
- Lista (IN and NOT IN)
- Combinação de *Strings* (LIKE and NOT LIKE)
- Valores desconhecidos (IS NULL e IS NOT NULL)
- Combinações destes (AND, OR)
- Negações (NOT)

```

SELECT *
      FROM authors
      WHERE zip > '90000'

```

5.10.2 ESCOLHA DE LINHAS BASEADA EM COMPARAÇÕES

```

SELECT select_list
      FROM table_list
      WHERE expression comparison_operator expression

```

Operadores de comparação:

- (=,>,<,<=,>=,<>,!<,>!, e !>)

```

SELECT au_lname, city
      FROM authors
      WHERE state = 'CA'

```

5.10.3 ESCOLHA DE LINHAS BASEADA EM AMPLITUDES

```

SELECT select_list

```

```
FROM table_list
WHERE expression [NOT] BETWEEN expression AND expression
```

```
SELECT pubdate, title
FROM titles
WHERE pubdate BETWEEN '1/1/91' AND '12/31/91'
```

5.10.4 ESCOLHA DE LINHAS BASEADA EM LISTAS

```
SELECT select_list
FROM table_list
WHERE expression [NOT] LIKE 'string'
```

Wildcard	Descrição
%	Qualquer string de zero ou mais caracteres
_	Qualquer caractere único
[]	Qualquer caractere único com amplitude ou set especificado
[^]	Qualquer caractere único com amplitude ou set não especificado

```
SELECT title, type
FROM titles
WHERE type IN ('mod_cook', 'trad_cook')
```

5.10.5 ESCOLHA DE LINHAS BASEADA EM VALORES DECONHECIDOS

```
SELECT select_list
FROM table_list
WHERE column_name IS [NOT] NULL
```

```
SELECT title
FROM titles
WHERE price IS NULL
```

5.10.6 ESCOLHA DE LINHAS BASEADA EM BUSCA DE VARIOS ARGUMENTOS

```
SELECT select_list
FROM table_list
WHERE [NOT] expression {AND|OR}[NOT] expression
```

```
SELECT title_id, title, pub_id, price, pubdate
FROM titles
WHERE (title LIKE 'T%' OR pub_id = '0877') AND
(price > $16.00)
```

5.10.7 ELIMINANDO DUPLICATAS

```
SELECT [ALL|DISTINCT] select_list
      FROM table_list
      WHERE search_conditions
```

```
SELECT DISTINCT city, state
      FROM authors
```

5.10.8 CLASSIFICANDO RESULTADOS

```
SELECT column_name [,column_name...]
      FROM table_list
      [ORDER BY column_name|select_list_number|expression
      [ASC|DESC][, column_name|select_list_number|expression
      [ASC|DESC].]
```

```
SELECT pub_id, type, price, title
      FROM titles
      ORDER BY type, price DESC
```

5.11 RECUPERAÇÃO DE DADOS - TÓPICOS AVANÇADOS

5.11.1 JOIN

```
SELECT column_name, column_name [,column_name...]
      FROM table_name, table_name [,table_name...]
      WHERE table_name, column_name, join_operator, table_name,
      column_name
```

Join operators:

- (=, >, <, <=, >=, <>, !=, !<, !>, =*, *=)
- *= → outer join

5.11.2 Natural JOIN

```
SELECT publishers.pub_id, publishers.pub_name,
      publishers.state, authors . *
      FROM publishers, authors
      WHERE publishers.city = authors.city
```


5.11.3 Eqüijoin

```
SELECT *
  FROM authors, publishers
 WHERE authors.city = publishers.city
```

5.11.4 JOINS com mais de duas Tabelas

```
SELECT stor_name, qty, title
  FROM titles, stores, sales
 WHERE titles.title_id = sales.title_id
 AND stores.stor_id = sales.stor_id
```

5.11.5 Auto JOINS

```
SELECT au1.au_fname, au.au_lname,
       au2.au_fname, au2.au_lname
  FROM authors au1, authors au2
 WHERE au1.city = 'Oakland'
 AND au1.sate = 'CA'
 AND au1.zip = au2.zip
 AND au1.au_id < au2.au_id
```

5.11.6 Outer JOINS

```
SELECT titles.title_id, title, qty
  FROM titles, sales
 WHERE titles.title_id *= sales.tilte_id
```

5.12 CRIANDO TRIGGERS

```
CREATE TRIGGER [owner.] trigger_name
  ON [owner.]table_name
  FOR {INSERT|UPDATE}
  AS
  IF UPDATE (column_name)...]
  [{AND|OR} UPDATE}
  sql_statements}
```

5.12.1 INSERT TRIGGER

```
CREATE TRIGGER loan_insert
  ON loan
  FOR INSERT
  AS
  UPDATE copy
```

```

SET on_loan = 'y'
FROM copy, inserted
WHERE copy.isbn = inserted.isbn
AND copy.copy_no = inserted.copy_no

```

5.12.2 DELETE TRIGGER

```

CREATE TRIGGER member_delete
ON member FOR DELETE
AS
    IF (SELECT COUNT (*)
        FROM loan, deleted
        WHERE loan.member_no = deleted.member_no) >
0
    BEGIN
        PRINT 'Transaction cannot be processed.'
        PRINT 'This member still has books on loan.'
        ROLLBACK TRANSACTION
    END
ELSE
    DELETE reservation
    FROM reservation, deleted
    WHERE reservation.member_no =
deleted.member_no

```

5.12.3 UPDATE TRIGGER

```

CREATE TRIGGER member_update
ON member
FOR UPDATE
AS
    IF UPDATE (member_no)
    BEGIN
        RAISEERROR (Transaction cannot be processed.\
***** Member number cannot be modified.', 10, 1)
        ROLLBACK TRANSACTION
    END

```

5.13 BULK COPY PROGRAM (BCP)

```

bcp [[database_name.]owner.]table_name {in|out}
datafile
[/m maxerrors] [/f formatfile] [/e errfile]
[/F firstrow] [/L lastrow] [/b batchsize]
[/n] [/c] [/E]
[/t field_term] [/r row_term]
[/i inputfile] [/o outputfile]
[/U login_id] [/P password] [/S servername] [/v] [/a
packet_size]

```

Exemplo:

```
bcp sau05..PROG in a:PROG.txt -U "usuário" -P  
"senha"  
-S graciosa
```

6. ACESSO VIA INTRANET / EXTRANET / INTERNET

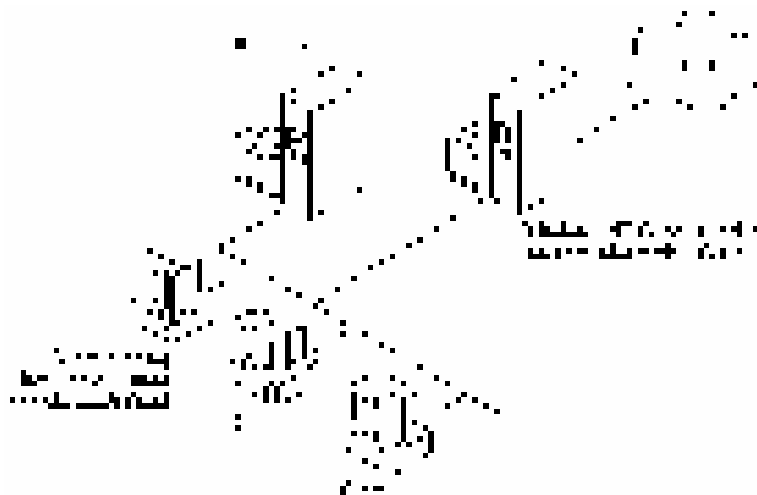
Acessar informações através da utilização de “navegadores”, seja no ambiente de uma Intranet, de uma Extranet ou da Internet, é uma tendência tecnológica, devido à facilidade de uso, e em muitos casos de implementação e facilidade de atualização, entre outras vantagens.

A Intranet é um ambiente interno à empresa, como exemplificado a seguir.



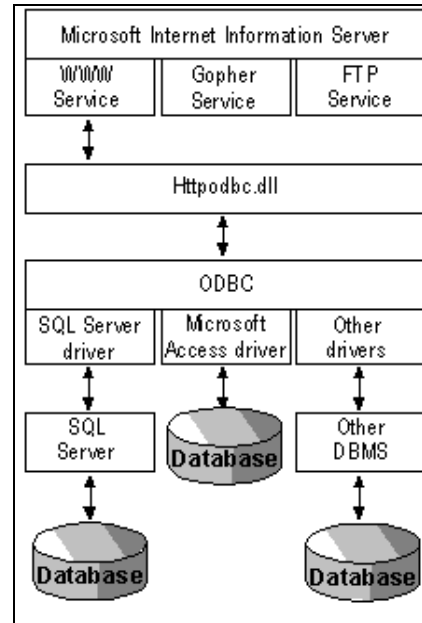
FONTE: Microsoft

Já no caso da Internet, o que muda é que os acessos serão permitidos a todo e qualquer usuário em qualquer parte do mundo, conforme exemplificado na figura a seguir.

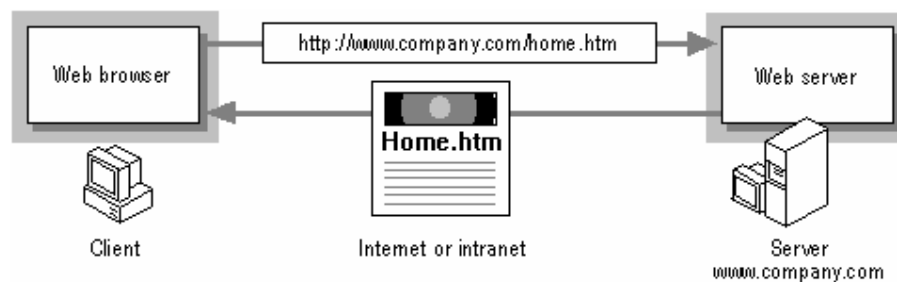


FONTE: Microsoft

Em ambos os casos utiliza-se um servidor dotado do sistema operacional *Windows NT* e acompanhado do *Microsoft Internet Information Server, IIS*, que é o servidor de serviços Internet (gerencia serviços de *ftp*, *gopher* e *www*). Nestes exemplos assumiu-se que o banco de dados que está disponível para os usuários, via *net*, é o *SQL Server*; mas na verdade qualquer outra ferramenta que suporte o protocolo *ODBC* poderá ser utilizada (Access, Sybase, Informix, Oracle, ...).



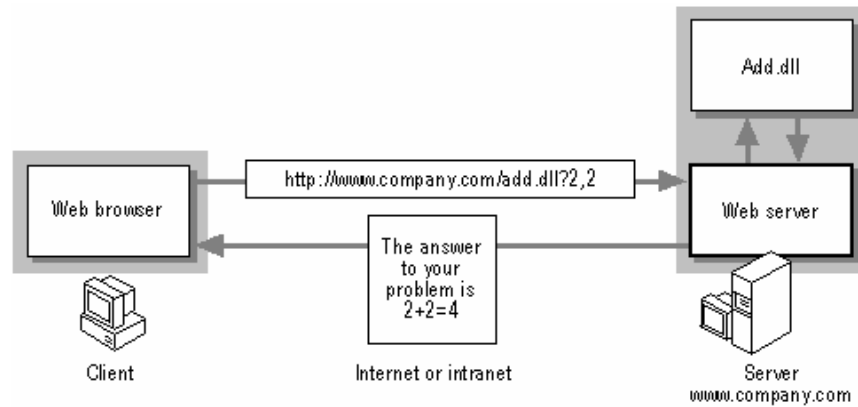
Interessa-nos em especial o serviço *www*, e o acesso a bancos de dados via protocolo *HTTP*. O acesso às informações contidas no servidor é feito de maneira relativamente simples. A partir da figura a seguir, veremos como isto é realizado.



FONTE: Microsoft

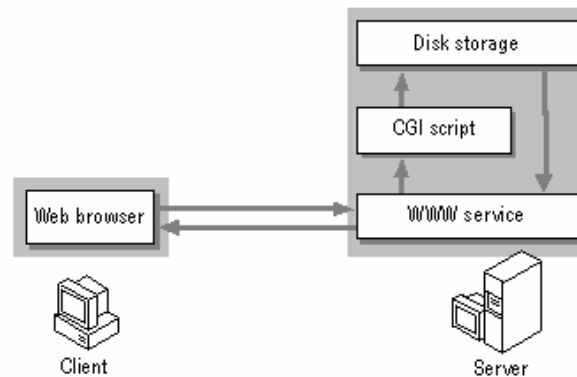
Como podemos observar, o navegador (*web browser*) comunica-se com o servidor (*web server*) utilizando o protocolo *HTTP*, o qual é portado no *TCP/IP*. O servidor, ao receber uma comunicação inicial envia como resposta uma seqüência *HTML*, através da qual o navegador efetua a formatação da página e mostra-a ao usuário.

Opcionalmente podem ser enviados ao servidor comandos adicionais, anexados ao endereço. Na figura a seguir exemplifica-se isto através do envio de um comando para execução da *library add.dll*, à qual serão passados dois argumentos



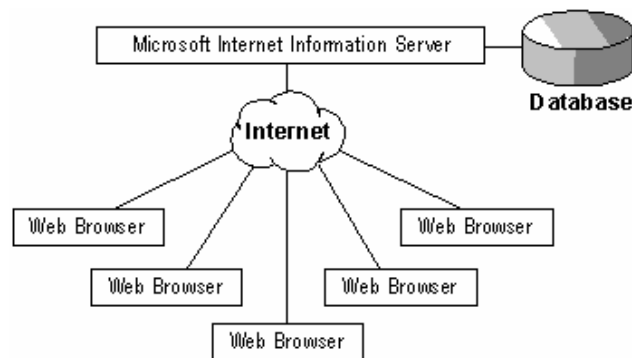
FONTE: Microsoft

O *Microsoft IIS* poderá ainda executar *scripts cgi*, bastante comuns em aplicações Internet.



FONTE: Microsoft

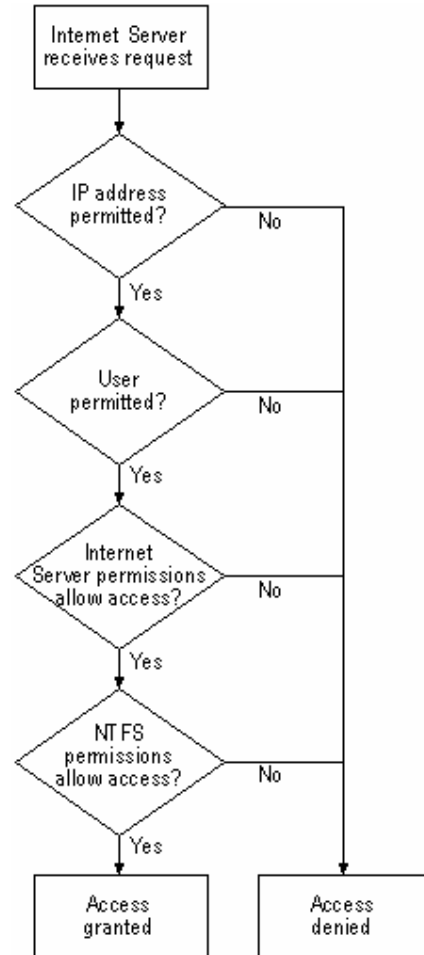
Para entendermos o que ocorre para que um usuário possa acessar informações em um banco de dados *SQL Server* (ou em outro que aceite conexões *ODBC*, como o *Access*), vamos basear-nos na figura a seguir.



FONTE: Microsoft

Todo o gerenciamento da comunicação com a Internet é efetuada pelo *IIS*. Para conectar-se a um banco de dados ele utiliza-se do *IDC*, *Internet Database Conector*, o qual é integrado ao *IIS* e efetua a conexão através do protocolo *ODBC*, possibilitando assim acesso a uma ampla gama de *databases*.

Antes de prosseguirmos, devemos ter em mente que é realizada uma checagem de segurança antes que comandos e/ou acesso sejam efetivamente executados, de maneira a manter a integridade e sigilo dos dados. A segurança do *IIS* é integrada à do *Windows NT*, deixando para este todo o gerenciamento de usuários, contas e direitos de acesso.



FONTE: Microsoft

6.1 EXEMPLO PRÁTICO

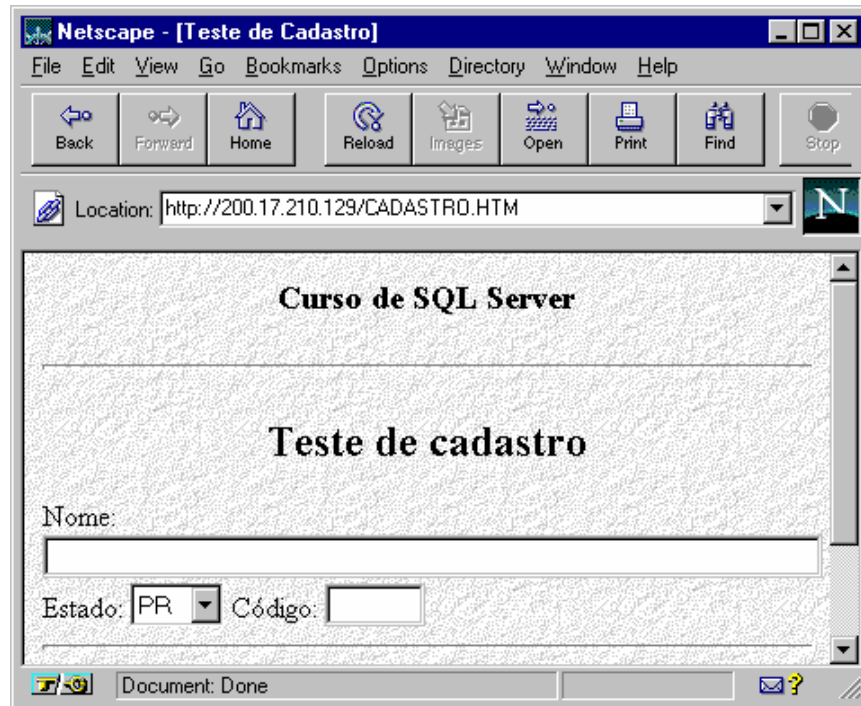
Vamos construir uma pequena aplicação de banco de dados, em que utilizaremos um *browser* como *front end*.

Nossa aplicação será formada por uma tabela, na qual poderemos cadastrar um nome, um estado e um código, conform a estrutura mostrada a seguir:

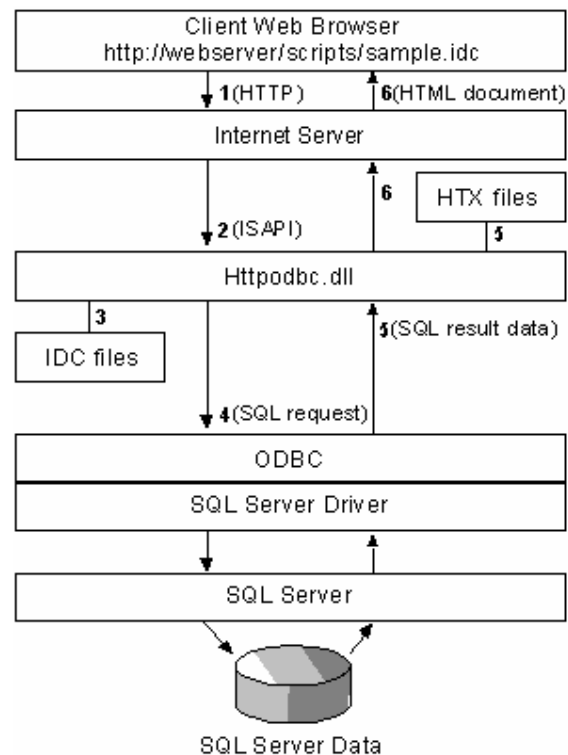
```

CREATE TABLE cadastro (
    numero int IDENTITY (1, 1) NOT NULL ,
    nome varchar (40) NULL ,
    estado char (2) NULL DEFAULT ('PR'),
    codigo int NOT NULL
)
  
```

Para acessar esta tabela simples, criaremos um acesso conforme mostrado a seguir; uma vez cadastrado, deveremos oferecer uma lista para consulta e possibilidades de alteração.

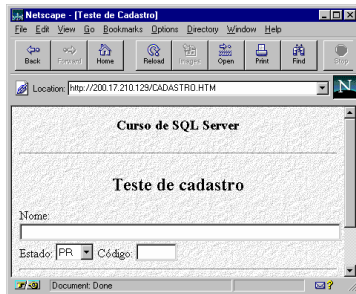


Quando as informações forem submetidas ao IIS, este irá realizar uma consulta no arquivo de conexão indicado pelo método *submit* do formulário, descobrindo então a qual banco de dados deverá se conectar. Uma vez conectado ao banco, será realizada a query passada pelo arquivo de conexão, que também passou os valores de campos recebidos do formulário. Realizada a consulta, o IIS irá utilizar o arquivo de modelo para montar uma seqüência de comandos *HTML* correspondentes à página que será enviada ao usuário. Desta maneira o *browser* enxergará *HTML* puro.



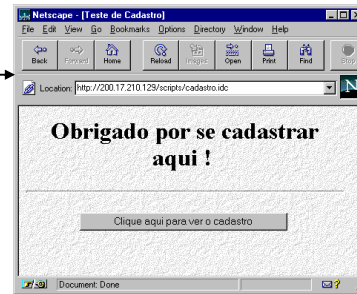
Vejamos como ficará nosso esquema de navegação:

Página INICIAL



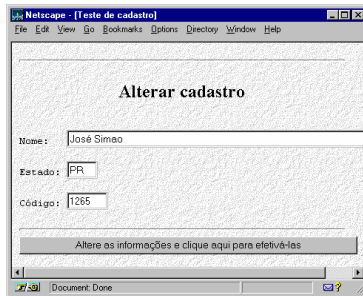
Arquivo: HTML

Agradecimento



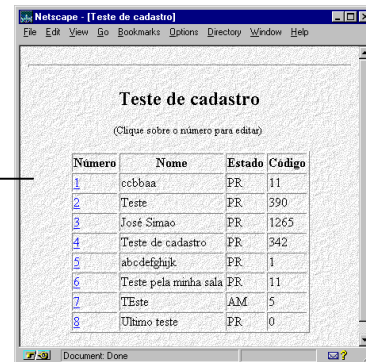
Arquivo: HTX

Tela para alteração dos dados



Arquivo: HTX

Lista



Arquivo: HTX

Teremos uma tela inicial, escrita em *HTML* padrão que conterá um *FORM*. Uma vez preenchido o formulário e submetido ao servidor, através do arquivo *IDC*, não mostrado acima, será realizada a inserção dos dados no database, e enviada uma tela de agradecimento ao usuário. Desta tela, o usuário terá possibilidade de conectar-se com o servidor para realizar uma consulta às informações cadastradas. Será novamente utilizado um arquivo *IDC*, o qual usará um novo arquivo de template, do tipo *HTX*, para enviar os dados (Lista) ao usuário. Nesta tela de resultados o usuário poderá escolher qualquer um dos itens existentes para proceder à sua alteração. O campo correspondente ao número será usado como chave de pesquisa, quando da alteração, mas não aparecerá na tela (deverá estar com o atributo de invisível).

6.2 ARQUIVOS NECESSÁRIOS E SCRIPTS

Utilizaremos os seguintes arquivos, cujo conteúdo será mostrado na seqüência:

Arquivo	Tipo	Finalidade
Cadastro	.htm	Tela inicial
Cadastro	.idc	Conexão para <i>INSERT</i> no <i>database</i>
Result	.htx	Mensagem de agradecimento
Todos	.idc	Conexão para <i>SELECT *</i> no <i>database</i>
Cadastro	.htx	Mostrar uma lista com o conteúdo do <i>database</i>
Cadpesq	.idc	Conexão para <i>SELECT WHERE</i> número = ?
Cadatu	.htx	Tela para alterações, com as informações atuais correspondentes ao número escolhido
Cadatu	.idc	Conexão para <i>UPDATE</i> no <i>database</i>

Note que estamos considerando apenas os arquivos básicos para a navegação e execução das tarefas, e que não será incluído nestes arquivos nenhum tipo de embelezamento, a não ser quanto a uma imagem de fundo, de maneira a deixar o código o mais inteligível possível.

6.2.1 Script para o arquivo cadastro.htm

```

<HTML>

<HEAD><TITLE>Teste de Cadastro</TITLE></HEAD>

<BODY BACKGROUND="/samples/images/backgrnd.gif">

<BODY BGCOLOR="FFFFFF">

<CENTER>

<H3>Curso de SQL Server</H3>

<HR>

<H2>Teste de cadastro</H2>

</CENTER>

<FORM METHOD="POST" ACTION="/scripts/cadastro.idc">

<P>

Nome:&nbsp;   <INPUT NAME="nome" VALUE="" size=60
maxlength=40><br>

Estado:&nbsp;   <SELECT NAME = "estado">

```

```

<OPTION VALUE = PR CHECKED>PR
<OPTION VALUE = SC >SC
<OPTION VALUE = RS >RS
<OPTION VALUE = SP >SP
<OPTION VALUE = AM >AM
<OPTION VALUE = PI >PI
<OPTION VALUE = MA >MA
<OPTION VALUE = BA >BA
<OPTION VALUE = RN >RN
<OPTION VALUE = MS >MS
<OPTION VALUE = TO >TO

</SELECT>

Código:&nbsp;<INPUT NAME="codigo" VALUE="" size=6
maxlength=4><br>

<HR>

<P>

<CENTER>

<INPUT TYPE="SUBMIT"
VALUE="Cadastrar">&nbsp;&nbsp;&nbsp;&nbsp;<INPUT TYPE="RESET"
VALUE="Limpar">

</CENTER>

</FORM>

</BODY>

</HTML>

```

6.2.2 Script para o arquivo cadastro.idc

```

Datasource: SRV-LAB1
Username: CADASTRO
Password: CADASTRO

```

```

Template: Result.htx
SQLStatement:
+INSERT cadastro..cadastro
+VALUES('%nome%', '%estado%', %codigo%)

```

6.2.3 Script para o arquivo result.htx

```

<HTML>
<HEAD>
<TITLE>
Teste de Cadastro
</TITLE>
</HEAD>
<BODY BACKGROUND="/samples/images/backgrnd.gif">
<BODY BGCOLOR="FFFFFF">
<CENTER>

<H1>
Obrigado por se cadastrar aqui !
</H1>
<HR>
<FORM ACTION="/scripts/Todos.idc" METHOD="POST">
<INPUT TYPE="SUBMIT" VALUE="Clique aqui para ver o
cadastro">
</FORM>
</CENTER>
</BODY>
</HTML>

```

6.2.4 Script para o arquivo todos.idc

```
Datasource: SRV-LAB1
```

```

Username: CADASTRO
Password: CADASTRO
Template: Cadastro.htx
SQLStatement:
+ SELECT * FROM CADASTRO..CADASTRO ORDER BY NUMERO

```

6.2.5 Script para o arquivo cadastro.htx

```

<HTML>
<HEAD><TITLE>Teste de cadastro</TITLE></HEAD>
<BODY BACKGROUND="/samples/images/backgrnd.gif">
<BODY BGCOLOR="FFFFFF">
<TABLE>
<HR>
<CENTER>
<H2>Teste de cadastro</H2>
<FONT SIZE = 2>
(Clique sobre o número para editar)
</FONT>
<P>
<TABLE BORDER>
<%begindetail%>
<%if CurrentRecord EQ 0 %>
<TR>
<TH><B>Número</B></TH><TH><B>Nome<BR></B></TH><TH><B>
Estado<BR></B></TH><TH><B>Código<BR></B></TH>
</TR>
<%endif%>
<TR>

```

```

        <TD><A
    HREF="/scripts/CadPesq.idc?proc=<%numero%>"><%numero%></A><
    /TD>

        <TD><%nome%></TD>

        <TD><%estado%></TD>

        <TD><%codigo%></TD>

    </TR>

    <%enddetail%>

    <P>

</TABLE>

</CENTER>

<P>

<%if CurrentRecord EQ 0 %>

<I><B>Não foi localizado nenhum</I></B>

<HR>

<%endif%>

</BODY>

</HTML>

```

6.2.6 Script para o arquivo cadpesq.idc

```

Datasource: SRV-LAB1

Username: CADASTRO

Password: CADASTRO

Template: CadAtu.htx

SQLStatement:

+SELECT * FROM CADASTRO..CADASTRO

+WHERE CADASTRO.NUMERO = %proc%

```

6.2.7 Script para o arquivo cadatu.htx

```

<HTML>

<HEAD><TITLE>Teste de cadastro</TITLE></HEAD>

```

```

<BODY BACKGROUND="/samples/images/backgrnd.gif">
<BODY BGCOLOR="FFFFFF">
<HR>
<CENTER>
<H2>Alterar cadastro</H2>
<P>
</CENTER>
<FORM ACTION = "/scripts/CadAtu.idc">
<%begindetail%>
    <INPUT TYPE = "HIDDEN" NAME="updnumero" VALUE =
<%numero%>><BR>
    <PRE>Nome:    <INPUT NAME="updnome" VALUE= "<%nome%>"
SIZE=60 MAXLENGTH=40></PRE>
    <PRE>Estado: <INPUT NAME="updestado" VALUE=
"<%estado%>" SIZE=4 MAXLENGTH=2></PRE>
    <PRE>Código: <INPUT NAME="updcodigo" VALUE=
<%codigo%> SIZE=6 MAXLENGTH=4></PRE>
<%enddetail%>
<P>
<HR>
    <INPUT TYPE="SUBMIT" VALUE="Altere as informações e
clique aqui para efetivá-las">
</BODY>
</HTML>

```

6.2.8 Script para o arquivo cadatu.idc

```

Datasource: SRV-LAB1
Username: CADASTRO
Password: CADASTRO
Template: Result.htx
SQLStatement:

```

```
+UPDATE cadastro..cadastro
+ SET   NOME = '%updnome%',
+   ESTADO = '%updestado%',
+   CODIGO = %updcodigo%
+ WHERE
+   NUMERO = %updnumero%
```