**SQL SERVER - ANOTAÇÕES**

**CRIANDO UM BANCO DE DADOS**

/\*

Cria o banco ‘Ecommerce’.

\*/

create database Ecommerce

**USANDO UM BANCO DE DADOS**

/\*

Passa a usar o banco ‘Ecommerce’.

\*/

use Ecommerce

**CRIANDO UMA TABELA**

/\*

Cria a tabela ‘Produtos’ com as colunas:

‘Codigo’ aceitando valores inteiros;

‘Nome’ aceitando valores alfanuméricos com no máximo 100

caracteres] e que aceita *nulo* como valor,

‘Descricao’ semelhante a varchar, porém, o SQL irá preencher os 10

caracteres, mesmo que não sejam todos usados. Por exemplo, uma

entrada ‘a’, resulta em ‘a ‘] e que não aceita valores *nulo*.

‘Preco’ aceitando valores decimais.

\*/

create table Produtos

(

Codigo int,

Nome varchar(100) null,

Descricao char(10) not null,

Preco float

)

/\*

Outra forma de criar uma tabela.

\*/

1. Clicar com o botão direito do mouse em: ‘Tabelas’ > Novo > Tabela.
2. Preencher as colunas com os dados desejados.
3. Clicar em ‘Salvar’ (Ctrl+S) e digitar o nome da tabela.

**DELETANDO UMA TABELA**

/\*

Deleta a tabela ‘Teste’ do banco de dados.

**OBS:** Para apagar apenas os valores da tabela, ver COMANDO DELETE.

\*/

drop table Teste

**COMANDOS SELECT**

/\*

Exibe os dados de todas as colunas da tabela ‘Clientes’.

**OBS:** É possível selecionar o nome da tabela e clicar

em ‘Alt+F1’ para ver os detalhes da mesma.

\*/

select \* from Clientes

/\*

Exibe os dados da coluna selecionada. Neste caso, a

coluna ‘codigo’ na tabela ‘Clientes’.

\*/

select codigo from Clientes

/\*

É possível selecionar e até repetir colunas.

\*/

select codigo, nome, codigo, \* from Clientes

/\*

Exibe todas as entradas onde ‘TipoPessoa’ for igual a ‘J’, na tabela Clientes.

\*/

select \*

from Clientes

where TipoPessoa = 'J'

/\*

Exibe todas as entradas onde ‘TipoPessoa’ for igual

a ‘J’ **E** ‘Codigo’ for igual a ‘1’, na tabela Clientes.

\*/

select \*

from Clientes

where Codigo = 1

**AND** TipoPessoa = 'J'

/\*

Exibe todas as entradas onde ‘TipoPessoa’ for igual

a ‘J’ **OU** ‘Codigo’ for igual a ‘1’, na tabela Clientes.

\*/

select \*

from Clientes

where Codigo = 1

**OU** TipoPessoa = 'J'

**COMANDO INSERT**

/\*

Modo padrão para inserir valores na tabela.

\*/

insert into Clientes (codigo, nome, tipoPessoa) values (1, 'Thiago', 'F');

/\*

É possível omitir o 'into'. Ao definir as colunas, é possível alterar

a ordem em que os elementos são inseridos na tabela.

\*/

insert Clientes (codigo, nome, tipoPessoa) values (2, 'Thiago', 'F');

/\*

Se não forem definidas as colunas, o SQL vai considerar que você está

usando todas as colunas, na ordem em que elas existem na tabela.

\*/

insert Clientes values (3, 'Thiago', 'F');

/\*

É possível fazer múltiplas entradas em uma única linha,

separando cada entrada por vírgula.

\*/

insert Clientes values (4, 'Thiago', 'F'), (5, 'Thiago', 'F');

**COMANDO UPDATE**

/\*

Atualiza o ‘Codigo’ para ‘7’ e o ‘Nome’ para ‘José’

em todas as entradas com ‘TipoPessoa’ igual a ‘J’.

\*/

update Clientes

set Codigo = 7,

Nome = 'José'

where TipoPessoa = 'J'

/\*

Sem a cláusula ‘where’, este comando irá atualizar

todas as entradas ‘TipoPessoa’ para ‘J’ em ‘Clientes’.

\*/

update Clientes

set TipoPessoa = 'J'

**COMANDO DELETE**

/\*

Deleta todos os registros da tabela ‘Clientes’.

\*/

delete

from Clientes

/\*

Deleta todos os registros onde ‘Codigo’ seja igual

a 5, 4, 3 e 2, da tabela ‘Clientes’.

\*/

delete

from Clientes

where Codigo in(5, 4, 3, 2)

**COMANDO ALTER**

/\*

Adiciona uma nova coluna (‘CodigoStatus’) a uma tabela existente (‘Pedido’).

\*/

alter table Pedido add CodigoStatus int

**CONVERTENDO ENTRADAS**

/\*

Na exibição da tabela ‘Pedido’, cria uma coluna onde todas as entradas

‘DataSolicitacao’ são convertidas para varchar, no padrão SQL.

\*/

select \*, convert(varchar, DataSolicitacao)

from Pedido

/\*

Na exibição da tabela ‘Pedido’, cria uma coluna onde todas as entradas

‘DataSolicitacao’ são convertidas para varchar, no padrão brasileiro.

\*/

select \*, convert(varchar, DataSolicitacao, 103)

from Pedido

**COMANDO CASE**

/\*

Na exibição da tabela ‘Clientes’, cria uma coluna onde todas as entradas

‘TipoPessoa’ igual a ‘J’ aparecem como ‘Juridica’, e ‘TipoPessoa’ igual a ‘F’,

aparecem como 'Física'.

\*/

select \*,

case

when TipoPessoa = ‘J’ then ‘Juridica’

when TipoPessoa = ‘F’ then ‘Física’

else ‘Pessoa indefinida’

end

from Clientes

/\*

Na exibição da tabela ‘Clientes’, cria uma coluna onde todas as entradas

‘TipoPessoa’ igual a ‘J’ aparecem como ‘Juridica’, e ‘TipoPessoa’ igual a ‘F’,

aparecem como 'Física', seguida da data no padrão brasileiro.

\*/

select \*,

case

when TipoPessoa = ‘J’ then ‘Juridica ’

when TipoPessoa = ‘F’ then ‘Física ’

else ‘Pessoa indefinida’

end + convert(varchar, GETDATE(), 103)

from Clientes

**SETANDO UMA CHAVE PRIMÁRIA**

/\*

Na tabela ‘Clientes’, adiciona a coluna ‘Codigo’ como a chave primária. Essa

regra impede que sejam inseridos novos dados com o mesmo valor de ‘Codigo’.

Por exemplo: se já existe um ‘Codigo’ igual a 1, é impossível inserir outro ‘Codigo’

igual a 1 na tabela.

\*/

alter table Clientes add constraint pk\_cliente primary key (Codigo)

/\*

Dependendo da aplicação, pode ser complicado lembrar quais são os

valores de ‘Codigo’ que já estão em uso. Uma alternativa, é setar a

coluna ‘Codigo’ como uma ‘Identidade’. Assim, o próprio SQL se encarrega

de incrementar o valor de ‘Codigo’ sempre que uma entrada é feita.

\*/

Clicar com o botão direito na tabela ‘Clientes’;

Clicar em ‘Design’;

Em ‘Especificação de Identidade’, alterar ‘(É identidade)’ para ‘Sim’.

