Estudo sobre Banco de dados SQL SERVER

Principais tipos de dados

Baseados em caracteres

|  |  |
| --- | --- |
| Tipo de dados | Descrição |
| Char | Armazena N caracteres fixos (até 8.000) no formato não Unicode. Independentemente da quantidade de caracteres utilizados, irá sempre armazenar o tamanho de caracteres do campo, sendo preenchido o restante com espaços em branco |
| Varchar | Armazena N caracteres (até 8.000) no formato não Unicode |
| Text | Armazena caracteres no formato não Unicode. Esse tipo de dado suporte até 2.147.483.647 caracteres e existem funções específicas para trabalhar com esse tipo de dado |

Baseados em numéricos inteiros

|  |  |
| --- | --- |
| Tipo de dados | Descrição |
| Smallint | Armazena valores numéricos inteiros, variando de -32.768 a 32.767 |
| Int | Armazena valores numéricos inteiros, variando de -2.147.483.648 a 2.147.483.647 |
| Bigint | Armazena valores numéricos inteiros, variando de -9.223.372.036.854.775.808 a -9.223.372.036.854.775.807 |
| Int | Armazena valores numéricos inteiros, variando de -2.147.483.648 a 2.147.483.647 |
| Money | Valores numéricos decimais variando de -922,337,203,685,477.5808 a +922,337,203,685,477.5807 |
| Numeric (N, N) | Armazena valores numéricos com casas decimais, utilizando precisão. O primeiro número entre os parênteses, representa a quantidade de inteiros a serem armazenados, o segundo número, indica a quantidade de casas decimais do número. |
| Real | Armazena valores numéricos aproximados com precisão de ponto flutuante, variando de -3.40E + 38 a 3.40E + 38 |

Baseados em data

|  |  |
| --- | --- |
| Tipo de dados | Descrição |
| Smalldatetime | Armazena data e hora, com precisão de minutos. |
| Datetime | Armazena data e hora, com precisão de centésimos de segundos. |
| Time | Armazena somente hora. Pode armazenas segundos até a fração de 99999999. |
| Date | Armazena somente data |
| Money | Valores numéricos decimais variando de -922,337,203,685,477.5808 a +922,337,203,685,477.5807 |
| Numeric (N, N) | Armazena valores numéricos com casas decimais, utilizando precisão. O primeiro número entre os parênteses, representa a quantidade de inteiros a serem armazenados, o segundo número, indica a quantidade de casas decimais do número. |
| Real | Armazena valores numéricos aproximados com precisão de ponto flutuante, variando de -3.40E + 38 a 3.40E + 38 |

Cláusulas SQL

Como uma frase, uma instrução SQL tem cláusulas. Cada cláusula executa uma função para a instrução SQL. Algumas cláusulas são necessárias em uma instrução SELECT. A tabela a seguir lista as cláusulas SQL mais comuns.

|  |  |
| --- | --- |
| Cláusula | O que ela faz |
| SELECT | Lista os campos de uma tabela que contêm dados de interesse |
| FROM | Lista as tabelas que contêm os campos listados na cláusula SELECT |
| WHERE | Especifica critérios de campo que devem ser atendidos por cada registro a ser incluído nos resultados. |
| Order by | Especifica como classificar os resultados. |
| Group by | Em uma SQL que contém funções agregadas, lista campos que não são resumidos na cláusula SELECT. |
| HAVING | Em uma SQL que contém funções agregadas, especifica condições que se aplicam aos campos resumidos na instrução SELECT. |

Organização da SQL

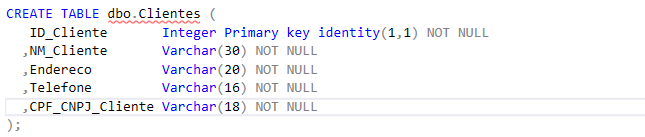
DDL - Linguagem de Definição de Dados - Define os comandos utilizados para criação (CREATE) de tabelas, views, índices, atualização dessas estruturas (ALTER), assim como a remoção (DROP);

CREATE

O comando CREATE é usado sempre que precisamos criar novos objetos na base de dados. Primeiramente, essa instrução pode ser utilizada para criar um banco completamente vazio, por exemplo:



Já utilizando o comando CREATE TABLE, podemos criar novas tabelas dentro de um banco de dados. Observe:



Repare que, ao criar a tabela, também especificamos quais colunas ela terá e os tipos de dados que cada coluna pode receber. Perceba ainda que, nas colunas de tipo VARCHAR, também foi especificado a quantidade máxima de caracteres que cada campo vai comportar.

Outra observação importante é a utilização da especificação PRIMARY KEY ao definirmos o campo id. Isso indica que essa coluna é uma chave-primária, ou seja, o seu conteúdo é único e representa a identificação do registro. A chave-primária é utilizada para facilitar a manipulação dos dados e para criar o relacionamento entre as tabelas.

ALTER

Por sua vez, o comando ALTER é utilizado quando precisamos alterar a estrutura de um objeto que já existe na base de dados. Um exemplo da aplicação dessa instrução é:



No exemplo mostrado, estamos adicionando a coluna “idade” na tabela “estudantes” e definindo que ela receberá dados do tipo INT. Além disso, o comando também pode ser usado da seguinte forma:



Nesse caso, estamos usando o comando ALTER TABLE junto com a instrução MODIFY para alterar o tipo de dados que a coluna “DT\_Nascimento” poderá receber.

DROP

O comando DROP é o responsável por deletar objetos da nossa base de dados. Utilizamos a sintaxe abaixo para realizar essa ação:



É interessante ressaltar que ele também pode ser usado para excluir o banco de dados completo. Para isso, inserimos a instrução dessa forma:



DQL - Linguagem de Consulta de Dados - Define o comando utilizado para que possamos consultar (SELECT) os dados armazenados no banco;

SELECT

É um dos comandos SQL mais importantes, pois com ele podemos elaborar diversas consultas aos registros da nossa base de dados. É possível, por exemplo, fazer uma pesquisa que retornará todos os campos de uma tabela dessa forma:



Perceba que, nessa consulta, o caractere \* é usado para indicar que os dados de todas as colunas da tabela “Clientes” devem ser retornados na pesquisa. Por isso, para fazer uma consulta mais específica, os campos desejados devem ser informados da seguinte maneira:



Além disso, utilizando a cláusula WHERE, podemos fazer uma consulta ainda mais refinada. Veja:



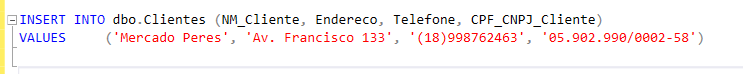
Como você deve ter notado, com o comando acima serão retornados apenas os nomes de estudantes que frequentam o curso de Desenvolvimento de Software. Ademais, caso queira uma lista ordenada, é possível utilizar a cláusula ORDER BY para que os dados sejam exibidos em ordem ascendente. Observe:



DML - Linguagem de Manipulação de Dados - Define os comandos utilizados para manipulação de dados no banco (INSERT, UPDATE e DELETE);

INSERT

O comando INSERT é usado para inserir dados em uma tabela do banco. Para realizar essa ação, podemos usar a seguinte sintaxe:



UPDATE

O comando UPDATE é usado para alterar dados em uma tabela do banco. Para realizar essa ação, podemos usar a seguinte sintaxe:



Na qual está alterando apenas um registro, ou seja, onde o ID\_Cliente for igual a 5.

Caso esse comando for utilizando sem cláusula WHERE, irá alterar todos os nomes de todos os clientes cadastrados na tabela, exemplo de sintaxe a não ser seguida:



DELETE

Por sua vez, o comando DELETE é utilizado para excluir informações da nossa base de dados. A sintaxe para realizar essa ação é bastante simples:



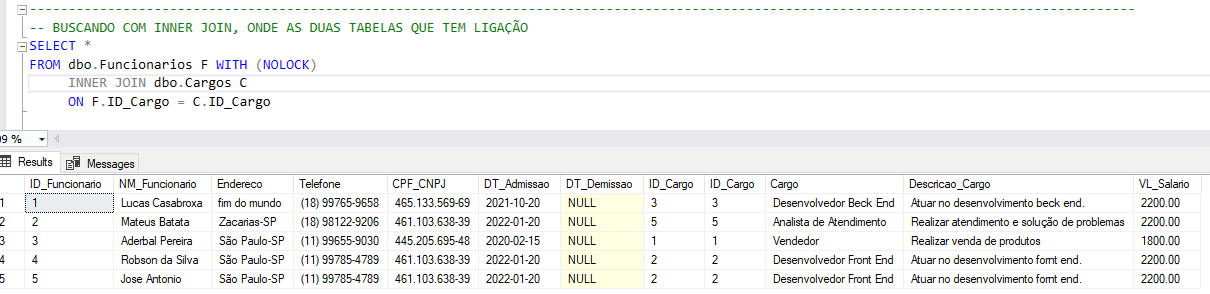
Na qual está excluindo apenas um registro, ou seja, onde o ID\_Cliente for igual a 5.

Caso esse comando for utilizando sem cláusula WHERE, irá excluir todos os nomes de todos os dados dos clientes cadastrados na tabela, exemplo de sintaxe a não ser seguida:

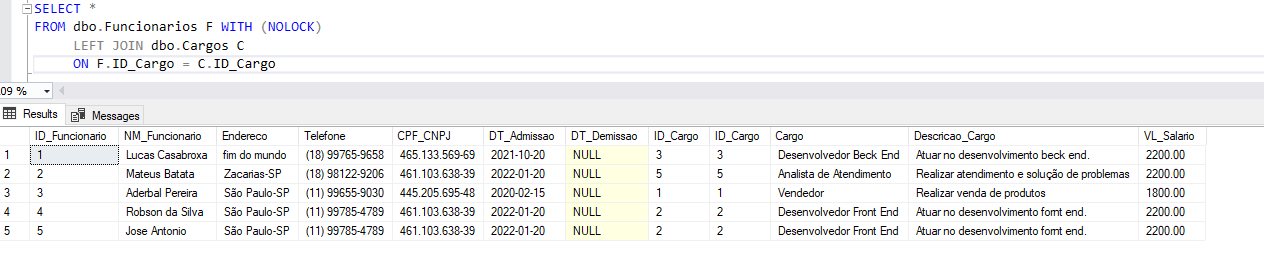


Cláusulas INNER JOIN, LEFT JOIN e RIGHT JOIN

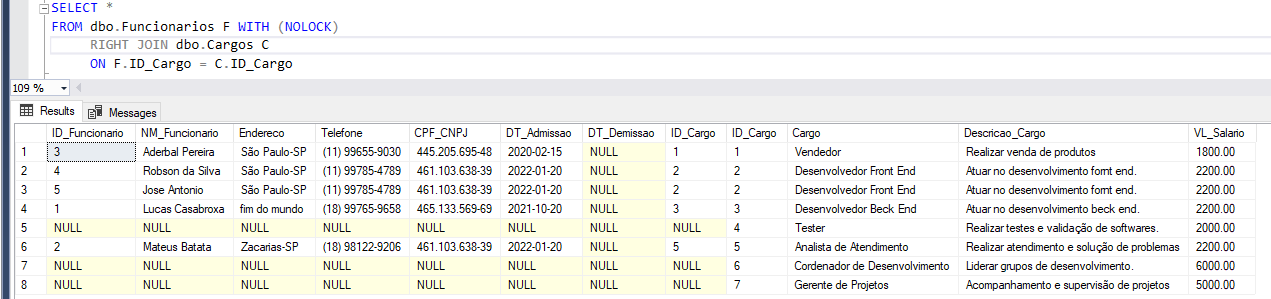
A cláusula INNER JOIN permite usar um operador de comparação para comparar os valores de colunas provenientes de tabelas associadas. Por meio desta cláusula, os registros de duas tabelas são usados para que sejam gerados os dados relacionados de ambas. Usamos as cláusulas WHERE e FROM para especificar esse tipo de associação.



A cláusula LEFT JOIN ou LEFT OUTER JOIN permite obter não apenas os dados relacionados de duas tabelas, mas também os dados não relacionados encontrados na tabela à esquerda da cláusula JOIN. Caso não existam dados relacionados entre as tabelas à esquerda e a direita do JOIN, os valores resultantes de todas as colunas da lista de seleção da tabela à direita serão nulos.



Ao contrário do LEFT JOIN, a cláusula RIGHT JOIN ou RIGHT OUTER JOIN retorna todos os dados encontrados na tabela à direita de JOIN. Caso não existam dados associados entre as tabelas à esquerda e à direita de JOIN, serão retornados valores nulos.

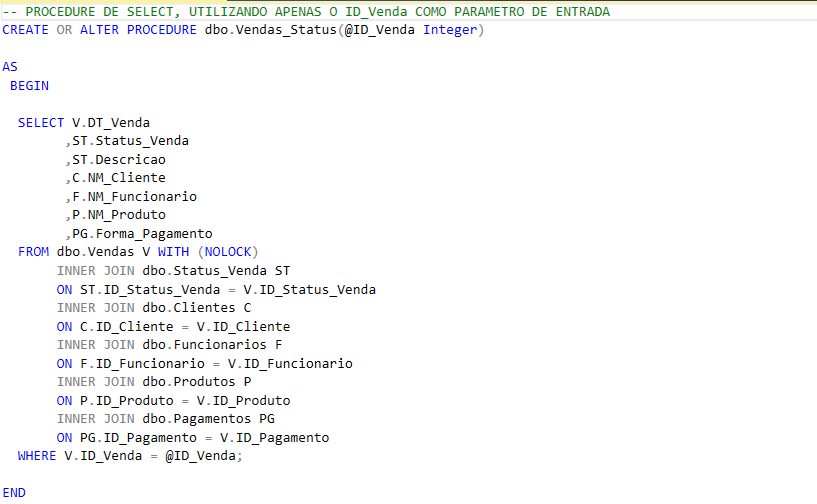


PROCEDURE

Uma procedure é um bloco de comandos ou instruções SQL organizados para executar uma ou mais tarefas. Ela pode ser utilizada para ser acionada através de uma chamada simples que executa uma série de outros comandos.

Exemplo:

A procedure abaixo é um select com várias tabelas, buscando através do parâmetro de entrada ID\_Venda.



Executando a procedure, passando ID\_Venda = 1, como parâmetro de entrada.



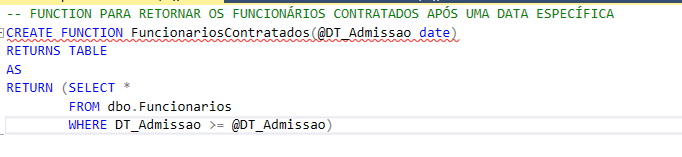
FUNCTION

Uma função definida pelo usuário é uma rotina Transact-SQL ou CLR (Common Language Runtime) que aceita parâmetros, executa uma ação, como um cálculo complexo, e retorna o resultado dessa ação como um valor. O valor de retorno pode ser um valor escalar (único) ou uma tabela.

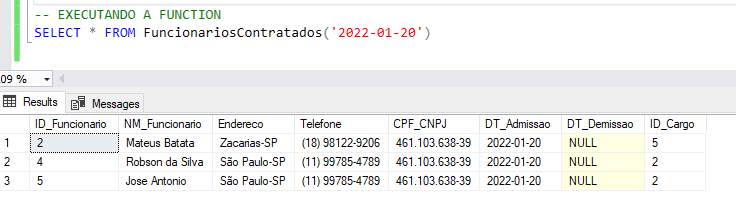
Exemplo:

A function abaixo é utilizada para retornar os funcionários contratados após uma data específica, essa data especifica é um parâmetro de entrada em que o usuário digita.

Exemplo: funcionários contratos na data 2022-01-20 ou superior



Executando a function, passando uma data, como parâmetro de entrada.



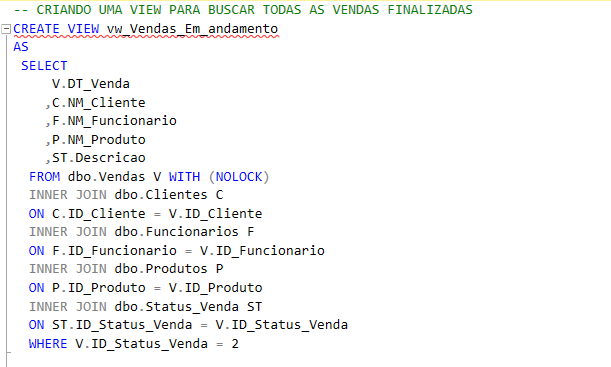
VIEW

VIEWS (ou visões em uma tradução livre) pode ser entendida como uma tabela virtual, composta pelo resultado de uma consulta a outras tabelas ou mesmo outras VIEWS.

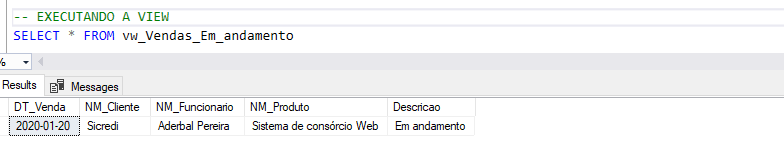
Ao contrário das tabelas normais, uma visão não faz parte fisicamente do banco, existindo apenas dinamicamente a partir das demais tabelas. Alterações dos dados nas tabelas refletem as alterações também na visão.

Podemos utiliza-las para executar um select muito grande, ao invés de ficar digitando toda vez, podemos deixar a view criada e dar um select nela.

Exemplo: A view abaixo é utilizada para retornar as vendas em que o ID\_Status\_Venda = 2, ou seja, em andamento.



Executando a view



Referências

<https://support.microsoft.com/pt-br/office/acesso-sql-conceitos-b%C3%A1sicos-vocabul%C3%A1rio-e-sintaxe-444d0303-cde1-424e-9a74-e8dc3e460671>

<https://blog.betrybe.com/desenvolvimento-web/comandos-sql/>