E-Book

.NET Developer

Banco de Dados

mensagem de recepção

Boas-vindas à nossa série de ebooks! Preparamos estes materiais pensando em como transmitir mais conhecimento diversificado para você explorar diferentes meios de desenvolver suas habilidades.

Aproveite este material e bons estudos!

SUMÁRIO

Introdução

Tipos de Banco de dados

Banco de dados relacional

O tipo mais usado atualmente, armazenando dados estruturados, sendo organizado em tabelas, com colunas e linhas, que se relacionam entre si.

Exemplos de Bancos de dados relacional :

Exemplo de tabela num Banco de dados Relacional:

Clientes					
Id	Nome	Sobrenome	Email	AceitaComunicados	DataCadastro
1	Leonardo	Buta	email@gmail.com	1	29/04/2022
2	Peter	Anderson	email@gmail.com	0	29/04/2022
3	Taylor	Adams	email@gmail.com	1	29/04/2022

- Nome da Tabela
  - Clientes

- Colunas
  - Id
  - Nome
  - Email
  - AceitarComunicados
  - DataCadastro

Banco de dados não relacional

Banco de dados onde os dados não são armazenados em tabelas, e sim armazenados de maneira **não estruturadas** ou s**emi-estruturadas.**

Exemplo de banco de dados não relacional:
Tipos de bancos de dado não relacional:

- document databases
- key-value databases
- wide-column stores
- graph databases

Exemplo de informação semi estruturada num banco de dados não relacional :

```
[
  {
    "Id": 1,
    "Nome_Produto": "Material de escritório",
    "Preco": "25.00",
    "DataVenda": "2022-04-23T01:23:26.9666539-03:00",
    "Desconto": null
  },
  {
    "Id": 2,
    "Nome_Produto": "Licença de Software",
    "Preco": 110.00,
    "DataVenda": "2022-04-23T01:23:26.9666539-03:00",
    "Desconto": 10,
    "Cupom": "1234"
  }
]
```

Percebe-se que existe uma estrutura, mas com diferenças entre as informações de "Id": 1 para a de "Id": 2.

Entendendo o DBMS

Database Management System, ou DBMS é um software utilizado para acessar, manipular e monitorar um sistema de banco de dados.

Exemplo de banco de dados com seu respectivo DBMS :

- SQL server
  - · SQL Server Management Studio

- MySQL
  - · MySQL Workbench

SQL, tabelas e tipos de dados

SQL e categorias de comandos

SQL ou Structured Query Language é uma linguagem de banco de dados padronizada, usada para consulta e manipulação de dados.

Categorias de comandos da linguagem SQL e comandos mais utilizados :

- DDL (Data Definition Language)
  - · CREATE

- DCL (Data Control Language)

- DML (Data Manipulation Language)
  - · INSERT
  - · UPDATE
  - · DELETE

- TCL (Transaction Control Language)

- DQL (Data Query Language)
  - · SELECT

Entendendo um database

Database é uma coleção de dados estruturados, agrupados de forma concisa. É composto de tabelas, procedures, views, etc. Um database pertence a um Servidor, que pode conter mais de um database.

Comando SELECT

```
SELECT * FROM Clientes
```

- SELECT *
  · Selecionar todos os dados de uma tabela, ou seja, todas as linhas e colunas
  · O caracter especial * significa todos os dados

- FROM clientes
  · ... da tabela clientes

Alguns incrementos do comando SELECT :

Comando	Função	Opção	Exemplos
ORDER BY	Ordenar a seleção	DESC	`ORDER BY Sobrenome DESC` `ORDER BY Nome, Sobrenome`

WHERE	Filtrar a seleção	AND, OR, LIKE	`WHERE Nome = 'Adam'`
			`AND Sobrenome = 'Reynolds'`
			`WHERE Nome LIKE 'G%'`
			`WHERE Nome LIKE '%G%'`

Comando INSERT

```
INSERT INTO Clientes
(Nome, Sobrenome, Email, AceitaComunicados, DataCadastro)
VALUES ('Leonardo', 'Buta', 'email@email.com', 1, GETDATE())
```

- INSERT INTO Clientes
  · Inserir na Tabela Clientes
  · O nome das colunas é opcional.

- VALUES
  ·Os valores devem corresponder à mesma ordem das colunas

Comando UPDATE

```
UPDATE Clientes
SET Email = 'emailatualizado@email.com'
WHERE Id = 1003
```

- SET
  - Comando para atualizar um campo com um novo valor

- WHERE
  - Indicar a linha da tabela identificada por um campo da coluna
  - Normalmente é utilizado com o campo Id
  - Não é recomendado utilizar o comando UPDATE sem WHERE

Comando DELETE

```
DELETE Clientes
WHERE Id = 1006
```

- WHERE
  - Normalmente é utilizado com o campo Id
  - Não é recomendado utilizar o comando DELETE sem WHERE

Comandos BEGIN TRAN e ROLLBACK

- BEGIN TRAN
  - Cria um ponto de restauração atual do database

- ROLLBACK
  - Restaura o database de acordo com a última execução do comando BEGIN TRAN

Tipos de dados mais utilizados

- Representar texto ou String Data Types
  - char(n)
  - varchar(n)

- Representar números ou Numeric Data Types
  - bit
  - int

- bigint
- decimal

- Representar datas e horários ou Date and Time Data Types
  - datetime2

Criação de tabela

```
CREATE TABLE Produtos (
    Id int IDENTITY(1,1) PRIMARY KEY NOT NULL,
    Nome varchar(255) NOT NULL,
    Cor varchar(50) NULL,
    Preco decimal(13, 2) NOT NULL,
    Tamanho varchar(5) NULL,
    Genero char(1) NULL
)
```

- Estrutura de criação da tabela
  - Nome da coluna
  - Tipo de dado da coluna
  - NULL ou NOT NULL

- IDENTITY
  - Para que o banco de dados gerencie o valor de Id
  - Começa com o valor de 1 e incrementa em +1 para cada novo registro

- PRIMARY KEY
  - Identifica a chave primária da tabela
  - Garante que o Id será único

Manipulando dados

Built-in functions

São funções pré-existentes que auxiliam na manipulação de dados, como por exemplo contar, somar,calcular média,etc...

COUNT

A função COUNT serve para contabilizar a quantidade de registros da tabela de uma maneira performática.

```
SELECT COUNT(*) QuantidadeProdutos FROM Produtos
```

- QuantidadeProdutos
  - Nome atribuído a coluna no resultado da função

A função COUNT também pode ser usada com o complemento WHERE :

```
SELECT COUNT(*) QuantidadeProdutosTamanhoM FROM Produtos WHERE Tamanho = 'M'
```

SUM

A função SUM aceita apenas as colunas cujos valores sejam do tipo numérico.

```
SELECT SUM(Preco) PrecoTotal FROM Produtos
```

A função SUM também pode ser usada com o complemento WHERE

```
SELECT SUM(Preco) PrecoTotalProdutosTamanhoM FROM Produtos WHERE Tamanho = 'M'
```

MIN, MAX e AVG

As funções MIN,MAX e AVG aceitam apenas as colunas cujos valores sejam do tipo numérico.

- MIN
  - Retorna o menor valor da coluna

- MAX
  - Retorna o maior valor da coluna

- AVG
  - Retorna a media dos valores da coluna

```
SELECT MIN(Preco) ProdutoMaisBaratoTamanhoM FROM Produtos WHERE Tamanho = 'M'

SELECT MAX(Preco) ProdutoMaisCaroTamanhoM FROM Produtos WHERE Tamanho = 'M'

SELECT AVG(Preco) FROM Produtos
```

Concatenando colunas

```
SELECT
    Nome + ', Cor: ' + Cor + ' - ' + Genero NomeProduto
FROM Produtos
```

- O sinal de + é utilizado para concatenar os valores
- No SQL a representação de uma String será sempre com aspas simples (‘)
- NomeProduto é o nome da coluna que aparecerá no resultado do SELECT

UPPER e LOWER

- UPPER

```
SELECT
    Nome + ', Cor: ' + Cor + ' - ' + Genero NomeProdutoCompleto,
    UPPER(Nome) Nome,
    LOWER(Cor) Cor
FROM Produtos
```

  - Retorna uma String transformando todos os caracteres em maiúsculo

- LOWER
  - Retorna uma String transformando todos os caracteres em minúsculo

GETDATE

Obter a data e hora atual do computador

```
UPDATE Produtos SET DataCadastro = GETDATE()
```

FORMAT

```
FORMAT(DataCadastro, 'dd/MM/yyyy HH:mm') Data
```

• A String de formatação do comando FORMAT é muito semelhante a formatação de datas no C#

Adicionando e removendo colunas por script

```
ALTER TABLE Produtos
ADD DataCadastro DATETIME2
```

• Cria uma nova coluna de nome DataCadastro na tabela Produtos

```
ALTER TABLE Produtos
DROP COLUMN DataCadastro
```

• Remove a coluna DataCadastro da tabela Produtos

Comando GROUP BY

Realiza um agrupamento de dados com base em uma determinada condição.

```
SELECT
    Tamanho,
    COUNT(*) Quantidade
FROM Produtos
WHERE Tamanho <> ''
GROUP BY Tamanho
ORDER BY Quantidade DESC
```

Nesse caso, a seleção será das colunas Tamanho e quantidade, agrupada pela coluna Tamanho. É importante utilizar a ordem correta dos comandos :

1. WHERE
    · Comando para instalar o pacote
2. GROUP BY
3. ORDER BY

Primary Key e Foreign Key

• Primary Key
    · Chave única que identifica cada registro na tabela

• Foreign Key
    · Chave que faz referência a uma Primary Key de outra tabela

Comando JOIN

Existem alguns tipos de comandos JOIN :

• INNER JOIN
• LEFT JOIN
• RIGHT JOIN
• FULL OUTER JOIN

O mais comum e utilizado e que será abordado pelo curso é o INNER JOIN.

INNER JOIN

Utilizado para juntar dados de duas tabelas e mostrar tudo num único resultado

```
SELECT
    Clientes.Nome,
    Clientes.Sobrenome,
    Clientes.Email,
    Enderecos.Rua,
    Enderecos.Bairro,
    Enderecos.Cidade,
    Enderecos.Estado
FROM
    Clientes
INNER JOIN Enderecos ON Clientes.Id = Enderecos.IdCliente
WHERE Clientes.Id = 4
```

III Constraints, Funções e Procedures

Constraints

Algumas Constraints já foram abordadas no curso. Exemplos de Constraints :

- NOT NULL
  - · Não permite valor nulo

- UNIQUE
  - · Valor único em toda a tabela

- CHECK
  - · Garante uma determinada condição

- DEFAULT
  - · Valor padrão para inserção

- PRIMARY KEY
  - · É uma combinação de NOT NULL e UNIQUE

- FOREIGN KEY
  - · Garante que um registro exista em outra tabela

UNIQUE

```
ALTER TABLE Produtos
ADD UNIQUE(Nome)
```

- Modifica a coluna “Nome” da tabela “Produtos” para conter apenas valores únicos, impedindo que a tabela contenha mais de um valor idêntico, garantindo que a coluna não tenha valores repetidos

CHECK

```
ALTER TABLE Produtos
ADD CONSTRAINT CHK_ColunaGenero CHECK(Genero = 'U' OR Genero = 'M' OR Genero = 'F')
```

• Modifica a tabela “Produtos” para que a coluna “Genero” aceite apenas Strings “U,M” ou “F”.

DEFAULT

```
ALTER TABLE Produtos
ADD DEFAULT GETDATE() FOR DataCadastro
```

• Modifica a coluna “DataCadastro” da tabela “Produtos” para que caso um novo registro seja adicionado omitindo a coluna “DataCadastro”, a mesma receba por padrão o valor da function GETDATE ao invés de nulo.

Apagando uma Constraint

```
ALTER TABLE Produtos
DROP CONSTRAINT UQ__Produtos__7D8FE3B2D9894E32
```

• Modifica a tabela “Produtos” removendo a Constraint UNIQUE de nome UQ_Produtos_7D8FE...

• DROP é o comando SQL utilizado para apagar dados armazenados

Stored Procedures

Procedures são utilizadas para substituir a utilização repetitiva de um determinado comando, diminuindo o trabalho de escrita. Parâmetros não são obrigatórios na criação de uma Procedure.

Criando uma Procedure

```
CREATE PROCEDURE InserirNovoProduto
@Nome varchar(255),
@Cor varchar(50),
@Preco decimal,
@Tamanho varchar(5),
@Genero char(1)

AS

INSERT INTO Produtos (Nome, Cor, Preco, Tamanho, Genero)
VALUES (@Nome, @Cor, @Preco, @Tamanho, @Genero)
```

• Criação de uma Procedure de nome “InserirNovoProduto” para ser utilizada no lugar do comando INSERT INTO, com a função de adicionar um novo registro na tabela “Produtos”

Utilizando uma Procedure

```
EXEC InserirNovoProduto
'NOVO PRODUTO PROCEDURE',
'COLORIDO',
50,
'G',
'U'
```

• EXEC ou EXECUTE é o comando de execução da procedure

• As linhas em sequência são os valores referentes às colunas na mesmo ordem da criação da Procedure

Functions

São parecidas com as Procedures, com a diferença de ser obrigatório conter um retorno na Function.

Criando uma Function

```
CREATE FUNCTION CalcularDesconto(@Preco DECIMAL(13, 2), @Porcentagem INT)
RETURNS DECIMAL(13, 2)

BEGIN
    RETURN @Preco - @Preco / 100 * @Porcentagem
END
```

- RETURNS
  - Diz respeito ao tipo de retorno que a função terá

- RETURN
  - O retorno em si da função

Utilizando uma Function

```
SELECT
    Nome,
    Preco,
    dbo.CalcularDesconto(Preco, 50) PrecoComDesconto
FROM Produtos WHERE Tamanho = 'M'
```

- Nesse caso, para executar uma Function, a utilização do “dbo.” antes do nome da função é obrigatório

Bancos não relacionais

Mongo DB

O Mongo DB é um banco de dados orientado a documentos, cujos dados não são armazenados em tabelas, e sim armazenados de maneira semi-estruturada no formato JSON.

O Mongo DB cria e gerencia automaticamente um "id" interno para cada novo documento da coleção, chamado de "_id".

Algumas diferenças de nomes entre o Mongo DB e o SQL Server:

SQL Server	MongoDB
Tabela	Coleção ou Collection
Linha/Registro	Documento ou Document

Comandos MongoSH :

Comando	Descrição
db.nome_da_colecao.insertOne({ JSON })	Cria um novo documento
db.nome_da_colecao.find({ condição JSON })	Faz uma busca nos documentos de acordo com a condição
db.nome_da_colecao.find({ }).sort({ condição 1 ou -1})	organiza em ordem crescente (1) ou decrescente (-1)
db.nome_da_colecao.updateOne({ condição },{ $set: { JSON } })	Atualiza um documento conforme a condição e o novo dado passado
db.nome_da_colecao.deleteOne({ condição })	Deleta um documento
db.nome_da_colecao.deleteMany({ condição })	Deleta mais de um documento
$lte	menor ou igual à

Comando	Descrição
$set	utilizado em conjunto do comando updateOne

Obrigado pela leitura!

Você não chegou aqui pulando páginas, né? 😏
Brincadeiras à parte, realmente nós da DIO esperamos que esteja curtindo sua jornada de aprendizado aqui conosco e desejamos seu sucesso sempre!
Vamos em frente!