# SQL Server

Readme.rmd

Sergio Pedro R Oliveira

2022-05-23

## Contents

**1 Objetivo** **2**

**2 Referência** **2**

**3 Modulo 24 - Instalação e delimitador GO** **3**
3.1 Instalação . . . 3
3.2 Acessando **SQL Server** pelo terminal . . . 4
3.3 Bancos do sistema . . . 4
3.4 Detalhes básicos da sintaxe do **SQL SERVER** . . . 5

**4 Modulo 25 - Arquitetura do SQL Server** **7**
4.1 Arquitetura do **SQL Server** na maquina . . . 7
4.2 Tipos de dados . . . 7
4.3 **TRANSACTION** - Transação . . . 8
4.4 Função **ERRO** . . . 8

**5 Andamento dos Estudos** **9**
5.1 Assunto em andamento . . . 9

# 1 Objetivo

Estudo dirigido de SQL Server.

# 2 Referência

Vídeo aulas “O curso completo de Banco de Dados e SQL, sem mistérios” - Udemy.

# 3 Modulo 24 - Instalação e delimitador GO

## 3.1 Instalação

### 3.1.1 Instalar SQL-server

- Versão:
  Versão usada é a express 2019, por ser a versão mais completa gratuita.

- Ubuntu
  https://docs.microsoft.com/pt-br/sql/linux/quickstart-install-connect-ubuntu?view=sql-server-ver15
  Basta seguir o passo a passo do site, ou pesquisar por pesquisar por "SQL Server Ubuntu" no youtube e seguir alguns tutoriais.

- Windows

### 3.1.2 Instalar Azure Data Studio

- Gerenciador de banco de dados usado para SQL-server, que estou usando no Ubuntu.

- Onde baixar:
  https://docs.microsoft.com/pt-br/sql/azure-data-studio/download-azure-data-studio?view=sql-server-ver15

## 3.2 Acessando SQL Server pelo terminal

- Execute o sqlcmd com parâmetros para o nome do SQL Server (-S), o nome de usuário (-U) e a senha (-P). Neste tutorial, você está se conectando localmente, portanto, o nome do servidor é localhost. O nome de usuário é SA (system administrator, equivalente ao root do MySQL) e a senha é a mesma fornecida para a conta SA durante a instalação.

sqlcmd -S localhost -U SA -P '*YourPassword*'

- É possível omitir a senha na linha de comando para receber uma solicitação para inseri-la.

sqlcmd -S localhost -U SA

## 3.3 Bancos do sistema

- São os bancos de dados do sistema que armazenam os dicionarios de dados.
- Bancos de dados do sistema:
  - **master**
    - É o banco de dados principal do sistema.
    - Todas as informações dos outros bancos de dados criados ficam armazenados nele.
  - **model**
    - São modelos de tabelas e bancos de dados, que ficam armazenados nesse banco de dados.
    - Pode servir de modelo automatico na criação de uma nova tabela ou banco de dados.
  - **msdb**
    - Armazenamento de rotinas.
    - Integrations Services, área de BI (ferramenta de **ETL**).
  - **tempdb**
    - Bancos de dados temporarios, ele é apagado todo vez que fecha e abre o sistema do banco de dados.
    - Muito utilizado para agilizar o teste de aplicações.

## 3.4 Detalhes básicos da sintaxe do SQL SERVER

### 3.4.1 Inserindo comentarios

- Um comentário é uma seqüência arbitrária de caracteres começando por dois hífens ("--") e prosseguindo até o fim da linha.

- Como alternativa, podem ser utilizados blocos de comentários no estilo C (/**bloco de comentarios**/). Utilizado para comentar mais de uma linha.

### 3.4.2 Extensão de arquivo script SQL

- O arquivo com o script SQL é salvo em ".sql".

- As três formas recomendadas de escrever os script's são:
  - **SQL server Management Studio** (**SSMS**)
    É um gerenciador de bancos de dados oferecidos pela microsoft, ótimo para gerenciar e trabalhar com banco de dados e arquivos ".sql".

  - **Azure**
    É um gerenciador de banco de dados e oferece ferramentas para o melhor entendimento e programação de um script ".sql".

  - Num arquivo de texto
    Preferencialmente o programa "**Sublime Text**", pois oferece a opção de escrever e salvar arquivos ".sql" com todas as ferramentas que envolve o processo.

### 3.4.3 Abrindo uma "nova consulta"

- Para começar a escrever um script no **SSMS** é necessario iniciar uma "nova consulta", abrir uma pagina que serve para escrever os comandos SQL.

- As duas formas de iniciar essa pagina são:
  - Clickar em nova consulta, parte superior da pagina.

  - Atalho **CRTL + N**

### 3.4.4 Uso do delimitador GO

- O **SQL Server** funciona da seguinte forma com seus script's:
  - Não precisa do delimitador para compilar o código, *processamento assincrono.*

  - Quando é pedido para compilar todo o script (sem seleções do código e sem uso de delimitador), o **SQL Server** executa o que for mais rapido primeiro, fora de ordem, por conta do *processamento assincrono.*

  - Ao selecionar uma parte do código ele compila apenas aquela parte do código.

  - Usando o delimitador **GO** executar o código por partes.

- Demilitador **GO**

  - O uso do GO ao final de cada instrução serve como delimitador.

  - O **GO** quebra o codigo em pequenos pacotes que são enviados para o servidor executar.

  - Colocando o **GO** no código ao final de cada instrução, o servidor não faz o *processamento assincrono*, assim quebrando o grande pacote que é o script inteiro, em pequenos pacotes para serem executados na ordem de envio.
- Modo de usar:

  - Colocar o **GO** ao final de cada instrução.

  - Inserir o **GO** na linha de baixo a instrução.

  - Sintaxe:
    **CREATE DATABASE** *nome_database*
    **GO**
    **USE** *nome_database*
    **GO**
    **CREATE TABLE** *nome_tabela*(
    *campo tipo*
    )
    **GO**

# 4 Modulo 25 - Arquitetura do SQL Server

## 4.1 Arquitetura do SQL Server na maquina

- No Ubuntu os dados de arquitetura ficam gravados no caminho:
  '/var/opt/mssql/data'

- No Windows os dados de arquitetura ficam gravados no caminho:
  '. . . /MSSQL/DATA'

## 4.2 Tipos de dados

- **MDF** (*master data file*)
  - Armazena dados do sistema (dicionario de dados).
  - Criação automatica pelo sistema.
  - Recomenda-se que use o MDF apenas para dados do sistema (mudança manual).
- **LDF** (*log data file*)
  - Armazena log's, transações, conjuntos de instruções.
  - Criação automatica pelo sistema.
  - É apagado quando explicitado (**BEGIN**) a transação, ao finalizada com **COMMIT** (confirmando a transação) ou **ROLLBACK** (desfazendo a transação).
- **NDF** (*not master data file*)
  - Não é criado automaticamente pelo sistema (criação manual), diferente dos outros.
  - Utilizado para armazenar dados.
  - Podendo armazenar dados atraves de grupos dados (*GP*), para melhor organizar os dados, assim fazendo a separação dos dados por assunto.

## 4.3 TRANSACTION - Transação

- É uma instrução que só executa as instruções dentro dela, no caso (**INSERT**, **UPDATE**, **DELETE**, ...), apenas se todas as instruções sejam concluidas com sucesso.

- Caso alguma instrução dentro dela dê ERRO, tudo é desfeito.

- Muito util para fazer operações de transação financeira entre contas.

  - Exemplo de transação financeira, transferencia de dinheiro entre contas:

    * Subtrair dinheiro de uma conta.

    * Somar dinheiro em outra conta.

- **COMMIT** ou **ROLLBACK**: Comandos que finalizam a transação onde o '**COMMIT**' confirma o conjunto de comandos e o '**ROLLBACK**' desfaz todo o processo executado pelo corpo de comandos caso tenha ocorrindo algum evento contrario ao desejado.

- Sintaxe:
  **BEGIN TRANSACTION** (ou **BEGIN**)
  UPDATE tabela SET coluna1_a_modificar = expressão1
  WHERE tabela IN (lista_dos_registros_a_modificar)
  UPDATE tabela SET coluna2_a_modificar = expressão2
  WHERE tabela IN (lista_dos_registros_a_modificar)
  **COMMIT** (ou **ROLLBACK**)

- Observação: Pode usar **BEGIN TRANSACTION** ou apenas **BEGIN**.

## 4.4 Função ERRO

- No **SQL Server** temos uma função de sistema que faz a endentificação de um erro dentro de uma transação chamada de '**@@ERROR**' função essa que por padrão recebe o valor 0 (zero) caso não ocorra nem um erro , no caso de algum erro ela assume o valor 1 (um).

- Uso da função '**@@ERROR**' dentro de um **IF**, para determinar uma *transação* (**TRANSACTION**) pode se mostrar uma boa solução.

- Sintaxe:
  **BEGIN TRANSACTION**
  **UPDATE FROM** *tabela*
  **SET** *campo_1* = 10.000
  **WHERE** *campo_1* < 50
  **IF @@ERROR** = 0
  **COMMIT**
  **ELSE**
  **ROLLBACK**
  **END**

# 5 Andamento dos Estudos

## 5.1 Assunto em andamento

Atualmente estou estudando Módulo 25.