**Manual do iniciante 2.0-SQL SERVER**

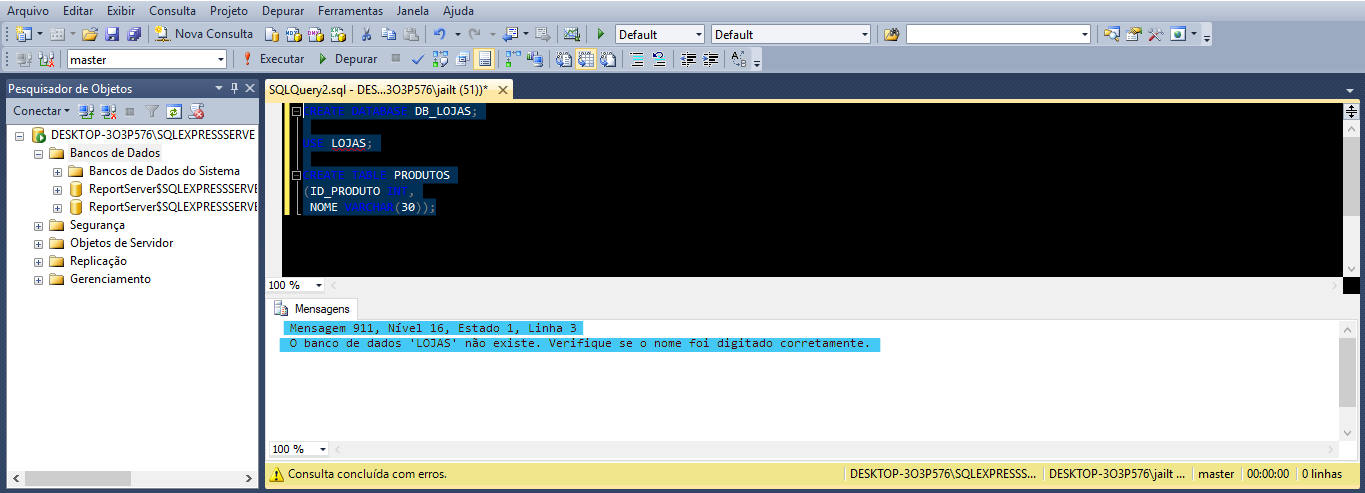
**Comandos do banco de dados**

**Comandos citados**

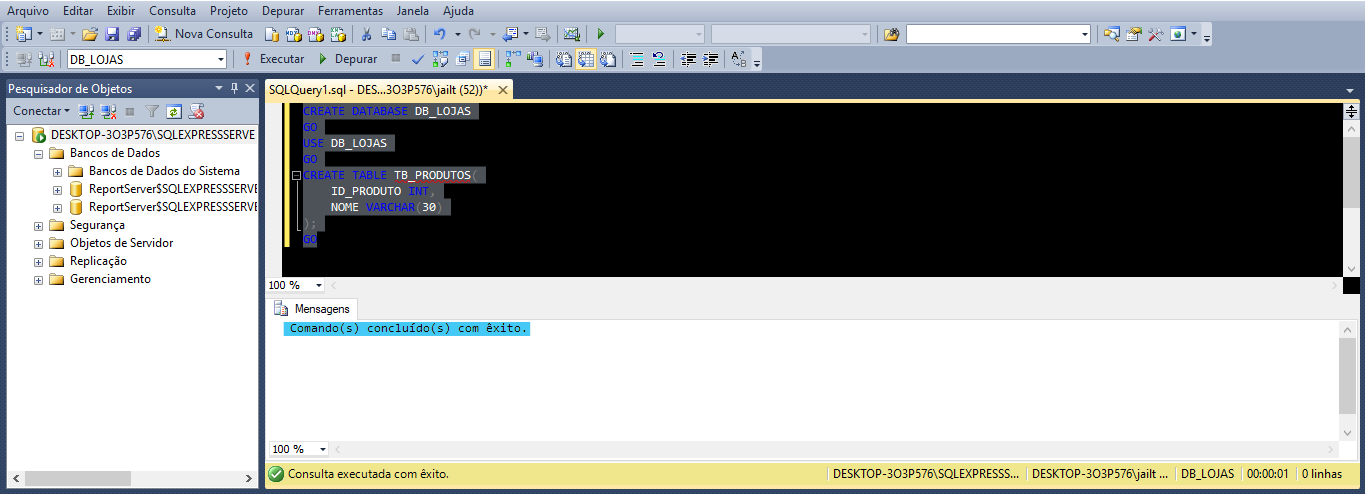
**Nomenclatura criada para teste**

***Clausula GO***

*No SQL SERVE é possível executar os comandos apenas selecionado o comando o qual deseja e clicando* ***F5*** *ou no botão , Quando existe vários códigos diferente sendo executado ao mesmo tempo o SQL SERVE executa do mais simples ao mais complexo ocasionando assim erros.*

***EX. CREATE DATABASE, USE, CREATE TABLE = USE, CREATE DATABASE, CREATE TABLE***

*Para não ocorrer o erro de ordem de execução é usado abaixo de cada comando a clausula* ***GO*** *assim executando o código na ordem que está.*

******

**TABELAS**

**Auto incremento na PRIMARY KEY**

Para adicionar o auto incremento na PRIMARY KEY no SQL SERVE faz utiliza código **IDENTITY** esse comando faz com que a coluna receba os valores automáticos.

**EX.** **CREATE TABLE TB\_ALUNOS(**

**ID\_ALUNO INT PRIMARY KEY IDENTITY,**

**NOME VARCHAR (30),**

**NASCIDO DATE,**

**SEXO CHAR (1),**

**CPF VARCHAR (11)**

**)**

**GO**

**CHECK**

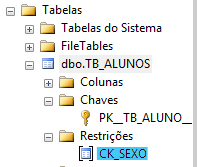
O **CHECK** é usado para determinar um padrão de dados que é inserido naquela tabela seguindo a mesma função que **ENUM** no MySQL, o CHECK pode ser adicionar na criação da tabela, mas por questão de organização é recomendado a criação dessa constraint após.

**EX.** **ALTER TABLE TB\_ALUNOS**

**ADD CONSTRAINT**

**CK\_SEXO CHECK (SEXO IN ('M','F'))**

**GO**



**NUMERIC**

O tipo **NUMERIC** é usado para declara números fracionados seguindo a mesma função do **FLOAT** no **MySQL** onde após declara o tipoé determinado a quantidade de casas e quantas serão após a virgula.

**CREATE TABLE TB\_PRECO (ID\_PRECO INT IDENTITY, VALOR NUMERIC (6,2))**

**INSERT**

Para usar o **INSERT** no **SQL SERVE** na coluna onde tem **IDETIFY** não é necessário informa o valor NULL como no MySQL apenas declara as colunas onde não seja **IDETIFY**.

**INSERT INTO TB\_ALUNOS**

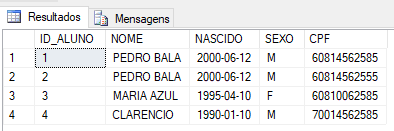
**VALUE**

**('PEDRO BALA','2000-06-12','M',60814562555),**

**('MARIA AZUL','1995-04-10','F',60810062585),**

**('CLARENCIO','1990-01-10','M',70014562585);**

**GO**



**CONSULTA**

**Consulta tabela**

Para detalhes da tabela usar comando

**SP\_COLUMNS TB\_ALUNOS**

**GO**

Ou

**SP\_HELP TB\_ALUNOS**

**GO**

Para ter mais detalhes da tabela.

**PROJEÇÃO**

**ISNULL**

Em colunas que o valor da célula é nulo para projetar um valor padrão é usado o **ISNULL** comparasse com comando **IFNULL** do **MySQL.**

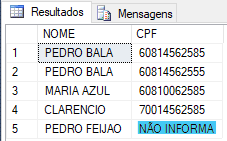
**SELECT**

**NOME,**

**ISNULL (CPF, 'NÃO INFORMADO') as "CPF"**

**FROM TB\_ALUNOS**

**GO**



**DATEDIFF**

Função **DATEDIFF (TIPO, DATA INICIAL, DATA FINAL)** é usada para calcular dias, meses ou anos de uma data outra.

**EX.** em um estoque é necessário saber quanto dias um produto passou fora da validade então usa o comando

**SELECT**

**NOME,**

**DATEDIFF (DAY, VALIDADE, GETDATE ()) AS "DIAS VENCIDO"**

**FROM TB\_ESTOQUE**

**GO**

Onde ele determinara a quantidade de dias entre a coluna **VALIDADE** até a data referência no caso **GETDATE ()** dia de hoje**,** Podendo substitui o valor de **DAY** por **MOUTH, YEAR** para saber meses e anos.

**DATENAME**

Para projeção do nome da data como STRING faz uso do comando **DATENAME ()**

**EX.**

**SELECT**

**NOME,**

**DATENAME(WEEKDAY,VALIDADE),**

**DATENAME(MONTH,VALIDADE)**

**FROM TB\_ESTOQUE**

**GO**



**DATAPART**

Para projeção a data como inteiro faz uso do comando **DATEPART ()**

**EX.**

**SELECT**

**NOME,**

**DATEPART (WEEKDAY, VALIDADE),**

**DATEPART (MONTH, VALIDADE)**

**FROM TB\_ESTOQUE**

**GO**



**DATEADD**

Para projeção a data futuras e passadas é feita através de **DATEADD (**TIPO, VALOR, ORIGEM**)** informado tipo se é dias, meses ou ano quanto tempo que deve contar e de qual coluna deve se basear.

**EX. nesse caso determinei que pegasse a data de dois anos atrás e definisse a coluna como data de produção.**

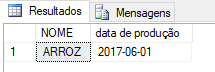
**SELECT**

**NOME,**

**DATEADD (YEAR, -2, VALIDADE) AS “DATA DE PRODUÇÃO”**

**FROM TB\_ESTOQUE**

**GO**



**CAST**

O CAST é usado para converter o tipo da coluna nas projeções assim o conteúdo se encaixe na futura conversão já que uma letra não pode ser convertida em número.

**SELECT CAST (VALOR AS VARCHAR) FROM TB\_PRODUTOS**

**GO**

**TRIGGER**

Criação da trigger no **SQL SERVE** segue um padrão onde são declaradas variáveis que receberam valores e essas variáveis iram ser inserida na tabela auxiliar, na trigger é definido como se fosse uma tabela o **INSERTED** e o **DELETED** para pegar valores antes e depois da alteração sofrida podendo também usar lógica de programação para definir regras onde casos não se encaixe a alteração não seja permitida.

**CREATE TRIGGER TG\_HIST\_BEM\_ATIVO**

**ON DBO. TB\_BENS\_ATIVOS \*1** tabela o qual deve.

**FOR UPDATE \*2**tipo de evento o qual deve monitora

**AS**

**DECLARE @NOME\_BEM VARCHAR(30) \*3**

**DECLARE @VALOR INT**

**DECLARE @USER VARCHAR(30)**

**DECLARE @DATA\_AUDIT INT**

**SELECT @NOME\_BEM = NOME, @VALOR=VALOR FROM DELETED \*4**

**SET @USER = SUSER\_NAME()**

**SET @DATA\_AUDIT = GETDATE()**

**\*5 INSERT INTO TB\_HIST\_BEM\_ATIVO VALUES(@NOME\_BEM, @VALOR,@USER, @DATA\_AUDIT)**

**GO**

**\*3** declaração de variáveis é feito através de DECLARE seguido de @ com nome da variável desejada e o tipo que deve ser igual da tabelas onde receber o dados e exportar.

**\*4** através da projeção do **SELECT** é atribuído os valores a variáveis onde é determinado **DELETED** para pegar os valores anteriores ou **INSERTED** para valores que foram inseridos.

**\*5** através do INSERT é atribuído valores para dentro da tabela os valores que a variáveis receberam.

Essa é a maneira básica da trigger porém elas podem ser usadas com lógica de programação

**--CRIAÇÃO DA TRIGGER**

**CREATE TRIGGER TG\_HISTORICO\_CARGO**

**ON DBO.TB\_MCEARA FOR UPDATE**

**AS**

**--DECLARADO VARIVEIS DAS ALTERAÇÕES**

**DECLARE @ID\_MILITAR INT**

**DECLARE @NOME VARCHAR(30)**

**DECLARE @NASCIDO DATE**

**DECLARE @INGRESSO DATE**

**DECLARE @DATA\_UP DATE**

**DECLARE @CARGO\_FK INT**

**DECLARE @ENDERECO\_FK INT**

**DECLARE @CONTATO\_FK INT**

**DECLARE @CPF VARCHAR(11)**

**--DECLARADO VARIAVEIS DE AUDITORIA**

**DECLARE @USER VARCHAR(30)**

**DECLARE @DATA DATETIME**

**SET @USER = SUSER\_NAME()**

**SET @DATA = GETDATE()**

**--VARIVEL DE COMPARAÇÃO**

**DECLARE @TEMPO\_I\_ANO INT**

**--ADERINDO VALORES AS VARIAVEIS 1**

**SELECT**

**@ID\_MILITAR = ID\_MILITAR,**

**@NOME = NOME,**

**@NASCIDO = NASCIDO,**

**@INGRESSO = INGRESSO,**

**@DATA\_UP = DATA\_UP,**

**@CARGO\_FK = CARGO\_FK,**

**@ENDERECO\_FK = ENDERECO\_FK,**

**@CONTATO\_FK = CONTATO\_FK,**

**@CPF = CPF**

**FROM DELETED**

**--ADERINDO VALORES A VARIAVEL DE COMPARAÇÃO**

**SELECT @TEMPO\_I\_ANO = CM.TEMPO\_I\_ANO**

**FROM TB\_CARGO\_MILITAR CM**

**WHERE CM.ID\_CARGO = @CARGO\_FK**

**--INSERINDO DADOS DE HISTORICO**

**INSERT INTO TB\_MCEARA\_HISTORIC VALUES**

**(**

**@ID\_MILITAR,**

**@NOME,**

**@NASCIDO,**

**@INGRESSO,**

**@DATA\_UP,**

**@CARGO\_FK,**

**@ENDERECO\_FK,**

**@CONTATO\_FK,**

**@CPF,**

**@USER,**

**@DATA**

**)**

**--DEFINIDO REGRA DE ROLLBACK**

**IF(@TEMPO\_I\_ANO > DATEDIFF(YEAR,@DATA\_UP,GETDATE()))**

**BEGIN**

**RAISERROR('ERRO DE INTERTICIO',16,2)**

**ROLLBACK TRANSACTION**

**END**

**--regra casos não cai na primeira regra**

**ELSE**

**BEGIN**

**PRINT('PROMOÇÃO ALCANÇADA')**

**END**

**GO**

**IFUPDATE**

**PROCEDURE**

**CREATE PROC NOME @VAR**

**AS**

**CONVERT(VARCHAR, @V\_DATA,105) – PORTUGUÊS**

**IF ELSE**

**CASE**

**WHILE**

**IF UPDATE(SALARIO)**