



Linguagem SQL

05 – Constraints e DML

Sand Onofre

Sand.Onofre@FaculdadeImpacta.com.br

Sumário

- Data Definition Language – DDL
 - Default
 - Check
- Data Manipulation Language – DML
 - Insert
 - Update
 - Delete
 - Truncate
- Exercícios

Data Definition Language - DEFAULT

Além dos tipos de dados, tamanhos, preenchimento obrigatório ou opcional (NULL / NOT NULL), auto numeração (IDENTITY), chave primária, única e estrangeira, podemos fazer uso de outros objetos.

A cláusula DEFAULT especifica o valor fornecido para a coluna quando um valor não for fornecido explicitamente durante uma inserção. Somente um valor constante, como uma cadeia de caracteres, números, datas, função de escalar ou NULL, pode ser usado como padrão.

As definições DEFAULT podem ser aplicadas a qualquer coluna, com exceção as colunas com a propriedade IDENTITY. Se um valor padrão for especificado para uma coluna de tipo definido pelo usuário, o tipo deverá oferecer suporte a uma conversão implícita.

DDL - DEFAULT

Não podem fazer referência a uma outra coluna da tabela, ou a outras tabelas, exibições ou procedimentos armazenados. As definições DEFAULT serão removidas quando a tabela for descartada.

<nome da coluna> <tipo de dados> CONSTRAINT <nome do default>
 DEFAULT (<valor, texto, data,
 função escalar>)

Exemplos:

MBAExterior VARCHAR(100) CONSTRAINT dfTextoNA DEFAULT 'Não'

Desconto DECIMAL(9, 2) CONSTRAINT dfDesconto DEFAULT 0

DataVenda DATE CONSTRAINT dfDataVenda DEFAULT (getdate())

DDL - DEFAULT

Exemplo:

CREATE TABLE Venda

```
(
  DataVenda date not null CONSTRAINT dfDataVenda DEFAULT (getdate())
, Quantidade smallint not null CONSTRAINT dfQtd DEFAULT (1)
, NumeroCliente int not null
, CONSTRAINT pkVenda PRIMARY KEY (DataVenda)
, CONSTRAINT fkVenda FOREIGN KEY (NumeroCliente)
  REFERENCES Cliente(idCliente)
);
```

Data Definition Language - CHECK

Os tipos de dados incluem uma restrição ao preenchimento dos dados em uma coluna, assim, quando definimos uma coluna como TINYINT, sabemos que os valores permitidos irão de 0 a 255.

No mesmo exemplo anterior poderíamos querer que os valores permitidos, além de serem numéricos, pudessem assumir somente os valores de 18 a 90. Esse tipo de restrição pode ser conseguida aplicando REGRAS, que é o significado da cláusula CHECK.

Uma coluna pode ter qualquer número de restrições CHECK e os critérios podem incluir diversas expressões lógicas combinadas com AND e OR. Várias restrições CHECK são validadas na ordem de criação.

DDL - CHECK

A avaliação do critério de pesquisa deve usar uma expressão Booleana (true/false) como base e não pode fazer referência a outra tabela.

A restrição CHECK no nível de coluna pode fazer referência somente à coluna restrita.

Restrições CHECK oferecem a mesma função de validação dos dados durante instruções INSERT e UPDATE.

Se existirem uma ou mais restrições CHECK para uma coluna, todas as restrições serão avaliadas.

`CONSTRAINT <nome da regra> CHECK (<coluna com expressão booleana>)`

DDL - CHECK

Exemplo:

CONSTRAINT ckIdade **CHECK** (Idade <= 100)

CONSTRAINT ckTaxa **CHECK** (Taxa >= 1 and Taxa <= 5)

CONSTRAINT CK_emp_id **CHECK** (emp_id **LIKE** '[A-Z][A-Z][A-Z][1-9][0-9][0-9][0-9][0-9][FM]' **OR** emp_id **LIKE** '[A-Z]-[A-Z][1-9][0-9][0-9][0-9][0-9][FM]')

CONSTRAINT CK_emp_id **CHECK** (emp_id IN ('1389', '0736', '0877', '1622', '1756') **OR** emp_id **LIKE** '99[0-9][0-9]')

DDL - CHECK

Exemplo:

create table Cliente

(

idCliente smallint identity(-32767, 1)

, Telefone VARCHAR(14)

, DataEntrada datetime

, Idade tinyint not null,

, constraint ckIdade CHECK (Idade between 18 and 90)

, constraint ckTelefone CHECK

(

Telefone LIKE '[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]' OR

Telefone LIKE '([0-9][0-9][0-9]) [0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]'

)

)

Data Definition Language – ALTER TABLE

As Constraints Primary key, Foreign Key, Unique, Default e Check, podem ser incluídas ou retiradas de uma tabela utilizando os mesmos comandos mencionados anteriormente. Exemplo:

```
ALTER TABLE Cliente Drop Constraint ckIdade
```

```
ALTER TABLE Cliente Add Constraint ckIdade CHECK (idade between 18 and 90)
```

```
ALTER TABLE Venda add constraint dfDataVenda DEFAULT (getdate()) for dataVenda
```

Data Manipulation Language

Após a definição de objetos que fazem a persistência de dados, precisamos de comandos SQL que manipulem informações dentro desses objetos.

As cláusulas a seguir tratam respectivamente de inserção, modificação e eliminação de registros dentro de tabelas:

INSERT

UPDATE

DELETE

Nos próximos slides iremos mostrar os comandos básicos para manipularmos informações.

Data Manipulation Language

INSERT

- A declaração **INSERT** adiciona uma ou mais linhas em uma tabela
- **INSERT** insere um ou mais valores (*data_values*) dentro (**INTO**) da tabela especificada (*table_or_view*)
- *column_list* é a lista de nome das colunas usadas para especificar as colunas das quais os dados são fornecidos

Sintaxe do **INSERT**:

```
INSERT [INTO] table_or_view [(column_list)] data_values
```

Manipulation Language

Declaração simples com INSERT

```
INSERT INTO MyTable (PriKey, Description)
VALUES (1, 'TPX450');
```

```
INSERT INTO Production.UnitMeasure
VALUES ('F2', 'Square Feet', GETDATE());
```

Inserindo Múltiplas linhas de Dados

```
INSERT INTO Production.UnitMeasure
VALUES ('F2', 'Square Feet', GETDATE()),
       ('Y2', 'Square Yards', GETDATE());
```

```
INSERT INTO MyTable (PriKey, Description)
VALUES (1, 'F200'), (2, 'GTX'), (3, 'CS');
```

Manipulation Language

INSERT usando VALUES

```
INSERT INTO MyTable (PriKey, Description)
VALUES (1, 'Texto 1')
```

```
INSERT INTO MyTable (PriKey, Description)
VALUES (1, 'F200'), (2, 'GTX'), (3, 'CS')
```

INSERT usando SELECT

```
INSERT INTO MyTable (PriKey, Description)
  SELECT ForeignKey, Description
  FROM SomeView
```

INSERT usando TOP (número de inserts)

```
INSERT TOP (#) INTO SomeTableA
  SELECT SomeColumnX, SomeColumnY
  FROM SomeTableB
```

Data Manipulation Language

Devemos lembrar que colunas com IDENTITY não devem ser mencionadas no INSERT, isso porque estas colunas são “administradas” pelo banco de dados e não pelos usuários.

Exemplo:

```
CREATE TABLE Veiculo
(
    idVeiculo INT IDENTITY NOT NULL
    , Placa char(8) NOT NULL
    , Marca varchar(20) NOT NULL
);
```

```
INSERT INTO Veiculo (Placa, Marca) VALUES ('XPT-7654', 'Ford');
INSERT INTO Veiculo (Marca, Placa) VALUES ('GM', 'KML-7299');
INSERT INTO Veiculo VALUES ('EXH-2566', 'Fiat');
```

Data Manipulation Language

- A declaração **DELETE** remove uma ou mais linhas numa tabela ou view
- **DELETE** remove linhas do parâmetro *table_or_view* que atender a condição no **WHERE** (*search condition*)
- O parâmetro *table_sources* pode ser usado para especificar tabelas ou views adicionais que podem ser usadas na cláusula **WHERE**

Sintaxe **DELETE**:

```
DELETE table_or_view  
  
FROM table_sources  
  
WHERE search_condition
```


Data Manipulation Language

DELETE sem a cláusula WHERE

```
DELETE FROM SomeTable;
```



```
DELETE FROM Sales.SalesPerson;
```

DELETE usando uma Subquery

```
DELETE FROM SomeTable  
WHERE SomeColumn IN  
    (Subquery Definition);
```



```
DELETE FROM  
Sales.SalesPersonQuotaHistory  
WHERE SalesPersonID IN  
    (SELECT SalesPersonID  
     FROM Sales.SalesPerson  
     WHERE SalesYTD >  
        2500000.00);
```

DELETE usando TOP

```
DELETE TOP (#) PERCENT  
FROM SomeTable;
```



```
DELETE TOP (2.5) PERCENT  
FROM  
Production.ProductInventory;
```

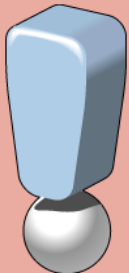
Data Manipulation Language

Sintaxe do TRUNCATE TABLE

```
TRUNCATE TABLE
    [ { database_name.[ schema_name ]. | schema_name . } ]
    table_name
[ ; ]
```

Exemplo do TRUNCATE TABLE

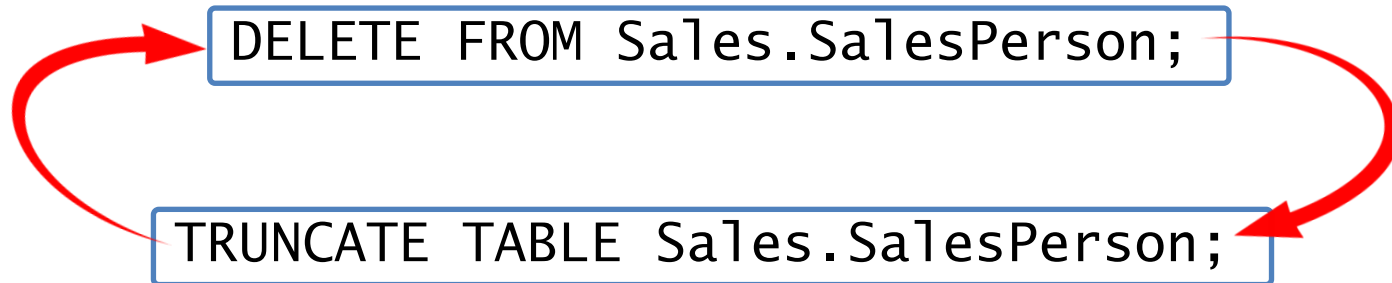
```
TRUNCATE TABLE Cliente;
```



1. Quando executado em uma tabela, reinicia a autonumeração (IDENTITY).
2. Não podemos usar TRUNCATE TABLE em tabelas referenciadas pela constraint FOREIGN KEY constraint.

Data Manipulation Language

- A declaração **TRUNCATE TABLE** é mais rápida que **DELETE**, mas não há como restringir as linhas que serão removidas através da cláusula **WHERE**, diferentemente do comando **DELETE**.



Data Manipulation Language

- Como boas práticas, primeiramente aplicamos o **SELECT** para verificar se os dados retornados são os que queremos eliminar

```
SELECT name FROM Cliente  
WHERE name like 'marcelo%';
```

- Caso o retorno seja realmente o que queremos eliminar, substituímos o **SELECT** pelo **DELETE**

```
DELETE FROM Cliente  
WHERE name like 'marcelo%';
```

Data Manipulation Language

- A declaração **UPDATE** altera valores dos dados de uma ou mais linhas de uma tabela
- Uma declaração **UPDATE** referenciando uma table or view pode alterar os dados somente em uma tabela ao mesmo tempo
- **UPDATE** tem três cláusulas principais:
 - SET – lista de campos, separados por vírgula, que serão alterados
 - FROM – fornece objetos fonte para a cláusula SET
 - WHERE – Especifica a condição de procura para aplicar as alterações com a cláusula SET

Sintaxe do UPDATE

UPDATE table_or_view

SET column_name = expression

FROM table_sources

WHERE search_condition

Data Manipulation Language

Declaração Simples do UPDATE

```
UPDATE SomeTable
SET Column = Value
```

```
UPDATE Sales.SalesPerson
SET Bonus = 6000;
```

```
UPDATE Sales.SalesPerson
SET Bonus = Bonus * 2;
```

UPDATE com a cláusula WHERE

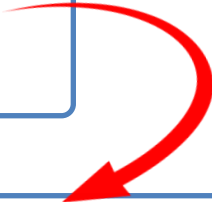
```
UPDATE SomeTable
SET Column = Value
WHERE SearchExpression
```

```
UPDATE Production.Product
SET Color = N'Metallic Red'
WHERE Name LIKE 'Road-250%'
AND Color = 'Red';
```

Data Manipulation Language

UPDATE usando uma Subquery

```
UPDATE SomeTable
SET Column = Value
FROM SomeSubquery
```




```
UPDATE Sales.SalesPerson
SET SalesYTD = SalesYTD + SubTotal
FROM Sales.SalesPerson AS sp
JOIN Sales.SalesOrderHeader AS so
    ON sp.BusinessEntityID = so.SalesPersonID
    AND so.OrderDate = (SELECT MAX(OrderDate)
                        FROM Sales.SalesOrderHeader
                        WHERE SalesPersonID =
                            sp.BusinessEntityID);
```

Antes

SalesYTD

677558.4653
4557045.0459

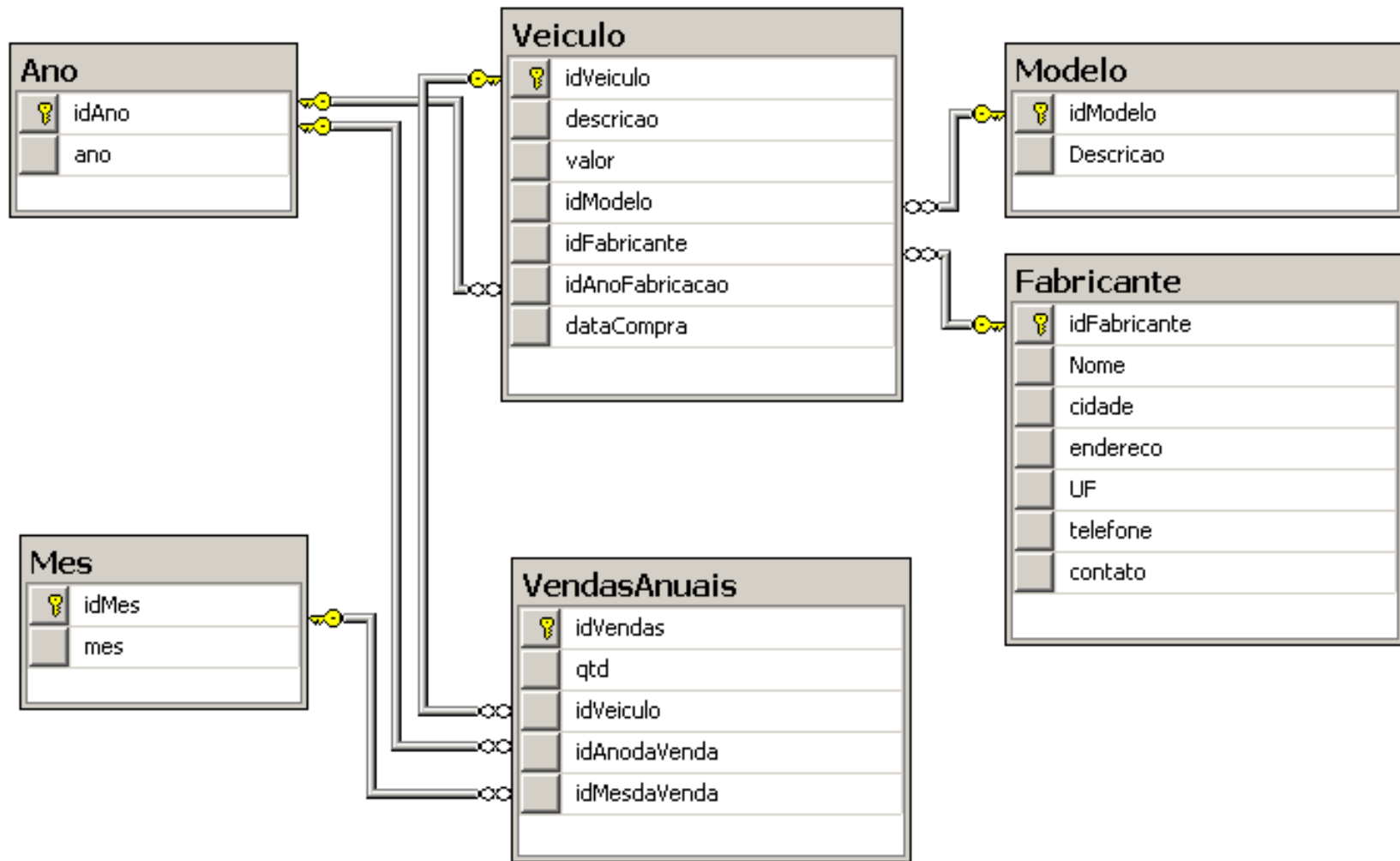
Depois



SalesYTD

721382.488
4593234.5123

Modelo Concessionaria



Cada aluno deverá se conectar com o ambiente em Nuvem, no database TEMP e rodar o script anexo na aula de hoje, referente a Carga Concessionária

Exercícios

Com os objetos criados por cada grupo, executar as seguintes modificações:

1. Altere as colunas DESCRICAO para um campo variável de 30 caracteres, obrigatório.
2. Insira o campo string variável de tamanho 20, de nome StatusVeiculo na tabela Veiculo.
3. Elimine a coluna CONTATO da tabela Fabricante.
4. Altere a tabela Veiculo para que a coluna StatusVeiculo tenha como valor padrão o texto ATIVO.
5. Altere a tabela Vendas Anuais para que a coluna QTD aceite valores entre 1 e 10000. (se houver valores fora desta faixa, usar a declaração DELETE para eliminar estes registros)
6. Insira os anos de 2016 até o ano corrente na tabela Ano.
7. Insira os modelos LST, KS e RS na tabela Modelo
8. Altere a tabela Mes, adicionando a coluna descricaoMes, de texto variável capaz de caber a maior das descrições dos meses (Janeiro, Fevereiro, ...).
9. Faça UPDATEs na tabela MES, colocando as descrições respectivas na coluna criada, de acordo com o respectivo valor apresentado na coluna MES.
10. Insira ao menos 3 Fabricantes (Triumph, KTM e Kymco) com informações reais para os campos da tabela.
11. Insira ao menos 3 registros (Tiger Explorer, ECX 450, DOWTOWN) na Tabela Veículo, com valores reais para os campos de valor, dataCompra e variando as informações de modelo, ano e Fabricante com os INSERTS que foram feitos anteriormente.
12. Para cada Veiculo inserido, insira ao menos 2 registros na tabela VendasAnuais para o Ano e Veículo inseridos, variando o mês e qtd.
13. Atualize o modelo LST para GTX.
14. Escolha um dos veículos inseridos e monte a sequência de comandos para eliminar o registro (mais de uma tabela), de forma que o banco de dados não gere erro na execução desta sequência de deleção.



Obrigado !

Sand Onofre
Sand.Onofre@FaculdadeImpacta.com.br