



Faculdade
IMPACTA
TECNOLOGIA



Linguagem SQL / Banco de Dados

Aula 06 – Projeto Físico:

DATA MANIPULATION LANGUAGE (DML)

Gustavo Bianchi Maia
gustavo.maia@faculdadeimpacta.com.br





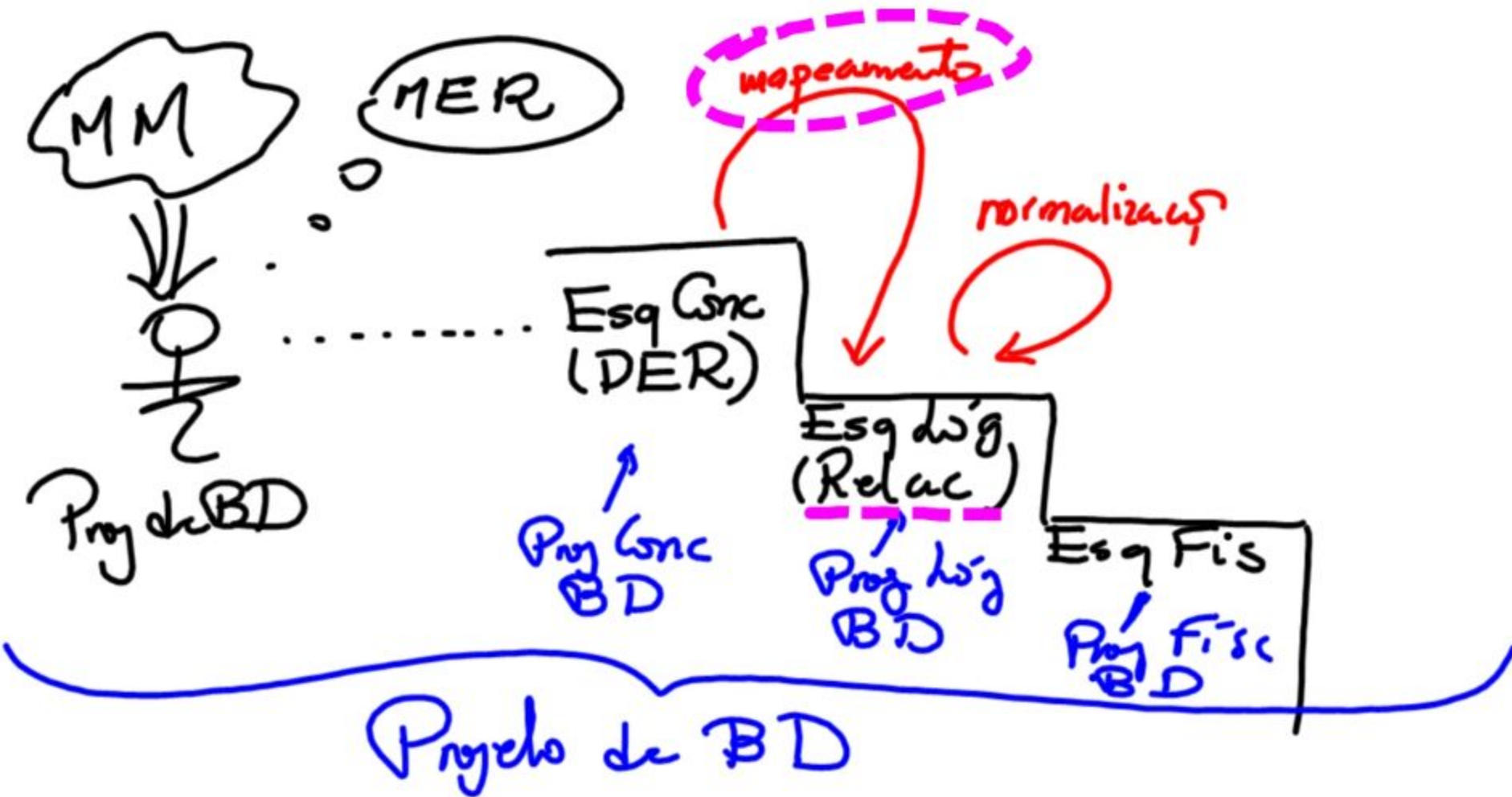
Agenda

- Revisão DDL (modelo físico)
- Sub-Linguagem DML
 - Insert
 - Delete
 - Update
- Exercícios





Modelo de Dados Relacional



Tipos de dados determinam quais os tipos de valores serão permitidos no armazenamento e os principais tipos são agrupados em categorias conforme mostrado abaixo:

Categorias dos Tipos de Dados	
Numéricos Exatos	Caractere Unicode
Numéricos Aproximados	Binários
Data e Hora	Outros Tipos
Strings de Caractere	





Data Definition Language

- Fator de Nulidade (NULL ou NOT NULL)
- Auto-preenchimento (valores auto-incrementais): IDENTITY (1,1)
- Criação da tabela

```
CREATE TABLE <nome da tabela>  
(  
    <nome coluna 1> <tipo da coluna> (<tamanho da coluna>) [NOT NULL] , ...  
);
```

- Regras:

- Primary Key

```
CONSTRAINT <nome da primary key> PRIMARY KEY (coluna1, coluna2, ...)
```

- Foreign Key

```
CONSTRAINT <nome da foreign key> FOREIGN KEY (coluna1, coluna2, ...)
```

```
REFERENCES <tabela da primary key> (coluna1, coluna2, ...)
```

- Unique

```
CONSTRAINT <nome da unique key> UNIQUE (coluna1, coluna2, ...)
```

- Check

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

- Default

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





Data Definition Language

CREATE TABLE Aluno

```
(  
  Matricula int not null IDENTITY (500, 1)  
  , Nome varchar(20)  
  , CONSTRAINT pkAluno  
    PRIMARY KEY (Matricula)  
);
```

Matricula	Nome
500	José
501	Pedro
502	Mario

CREATE TABLE Prova

```
(  
  idProva int NOT NULL IDENTITY (1, 1)  
  , Matricula int NOT NULL  
  , Nota decimal(4,2) NOT NULL  
  , CONSTRAINT pkProva PRIMARY KEY (idProva)  
  , CONSTRAINT fkProva FOREIGN KEY (Matricula)  
    REFERENCES Aluno(Matricula)  
);
```

idProva	Matricula	Nota
1	500	9
2	500	8
3	502	7
4	502	3
5	502	1



Imposições de preenchimento:

- PK → Uma por tabela, dados únicos, não aceita NULOS
- FK → Tipo de dado da PK, só aceita dados já utilizados na PK (colunas podem ser PK e FK)
- UQ → dados únicos, aceita NULOS
- DF → preenche com dados em substituição ao NULO, um por coluna
- CK → regra de validação booleana (true / false), aceita QQ código SQL para
- Identity → Uma por tabela, campos cujo preenchimento é controlado pelo sistema
- Campos calculados → Fórmulas, não são colunas que recebem dados ou valores

Sintaxe:

- Nomes com espaço ou palavras reservadas devem usar colchetes, ex: [nome com espaço]

Ordem dos comandos:

- Ordem de criação (CREATE) das tabelas (PK → FK)
- Ordem de remoção (DROP) das tabelas (FK → PK)
- Ordem de alteração estrutural da tabela (ALTER)

Remove dependências (constraints)

Remove dependências (colunas)

Realiza a alteração

Reconstrói dependências (colunas)

Reconstrói dependências (constraints).





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.





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
```



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');
```





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 (1) INTO SomeTableA  
  SELECT SomeColumnX, SomeColumnY  
  FROM SomeTableB
```



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(1,1) NOT NULL
```

```
  , Placa AS char(8) NOT NULL
```

```
  , Marca AS 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');
```





Para inspecionar os dados recém inseridos:

```
Select * from Veiculo
```

Para descobrir o ID do último veículo recém inserido

```
Select @@identity
```

```
Select SCOPE_IDENTITY()
```

```
INSERT INTO Veiculo (Placa, Marca) OUTPUT inserted.* VALUES ( 'XPT-7654', 'Ford');
```

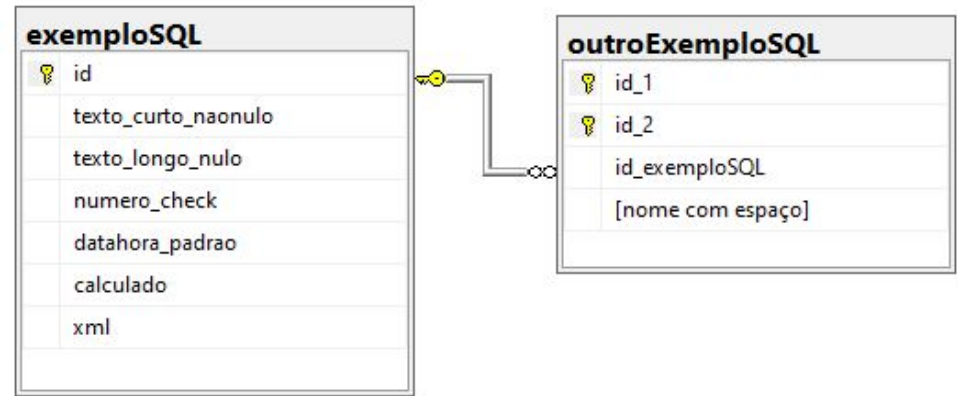




Data Manipulation Language

Erros comuns que devem ser evitados:

Seja a seguinte estrutura de exemplo:



```
CREATE TABLE exemplosql
(
    id                INT NOT NULL IDENTITY(1, 1),
    texto_curto_naonulo VARCHAR(5) NOT NULL,
    texto_longo_nulo   NVARCHAR(max) NULL,
    numero_check       INT NOT NULL CHECK (numero_check >= 0 ),
    datahora_padrao    DATETIME NOT NULL DEFAULT( Getdate() ),
    calculado          AS ( numero_check * 10000 ),
    xml                XML NULL,
    CONSTRAINT uq_exemplosql_texto_curto_naonulo UNIQUE (texto_curto_naonulo),
    CONSTRAINT pk_exemplosql PRIMARY KEY ( id )
)

CREATE TABLE outroexemplosql
(
    id_1              INT NOT NULL IDENTITY(1, 1),
    id_2              VARCHAR(50) NOT NULL,
    id_exemplosql     INT NOT NULL,
    [nome com espaço] VARCHAR(50) NULL,
    CONSTRAINT pk_outroexemplosql PRIMARY KEY(id_1, id_2),
    CONSTRAINT fk_outroexemplosql_id FOREIGN KEY ( id_exemplosql )
        REFERENCES exemplosql(id)
)
```





Erros comuns que devem ser evitados:

Não fornecer uma coluna obrigatória:

```
INSERT INTO exemplosql (texto_curto_naonulo,texto_longo_nulo) VALUES ( 'abc', 'abcd' )
```

Não é possível inserir o valor NULL na coluna 'numero_check', tabela 'DatabaseName.dbo.exemploSQL'; a coluna não permite nulos. Falha em INSERT.

Exceder o tamanho de um campo:

```
INSERT INTO exemplosql (texto_curto_naonulo,numero_check) VALUES ( 'abcdefg', 10 )
```

Os dados de sequência ou binários estão truncados na tabela 'DatabaseName.dbo.exemploSQL', coluna 'texto_curto_naonulo'. Valor truncado: 'abcde'.

Tipos incompatíveis:

```
INSERT INTO exemplosql (texto_curto_naonulo, numero_check) VALUES ( 'abcd', 'dez' )
```

Falha ao converter o varchar valor 'dez' para o tipo de dados int.

Tentativa de inserção em campo auto incremental:

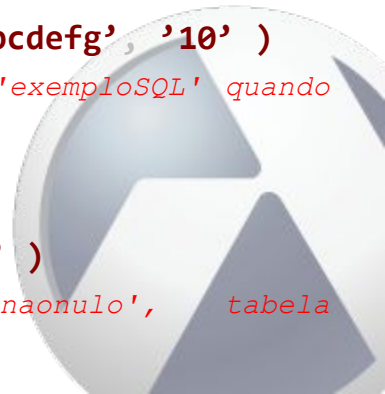
```
INSERT INTO exemplosql (id, texto_curto_naonulo, numero_check) VALUES ( 1, 'abcdefg', '10' )
```

Não é possível inserir um valor explícito para a coluna de identidade na tabela 'exemploSQL' quando IDENTITY_INSERT está definido como OFF.

Inserção de nulos em colunas que não aceitam nulos:

```
INSERT INTO exemplosql (texto_curto_naonulo, numero_check) VALUES ( NULL, '10' )
```

Não é possível inserir o valor NULL na coluna 'texto_curto_naonulo', tabela 'DatabaseName.dbo.exemploSQL'; a coluna não permite nulos. Falha em INSERT.





Erros comuns que devem ser evitados:

Valores não passaram no teste de validação (check) :

```
INSERT INTO exemplosql (texto_curto_naonulo, numero_check) VALUES ( 'abcd', -1 )
```

A instrução INSERT conflitou com a restrição do CHECK "CK_exemploSQ_numer__74AE54BC". O conflito ocorreu no banco de dados "DatabaseName", tabela "dbo.exemploSQL", column 'numero_check'.

Violação de constraint de unicidade :

```
INSERT INTO exemplosql (texto_curto_naonulo, numero_check) VALUES ( 'abcd', 10 )
```

(1 linha afetada)

```
INSERT INTO exemplosql (texto_curto_naonulo, numero_check) VALUES ( 'abcd', 11 )
```

Violação da restrição UNIQUE KEY 'UQ_exemploSQL_texto_curto_naonulo'. Não é possível inserir a chave duplicada no objeto 'dbo.exemploSQL'. O valor de chave duplicada é (abcd).

Violação de constraint de chave primária na tabela dependente:

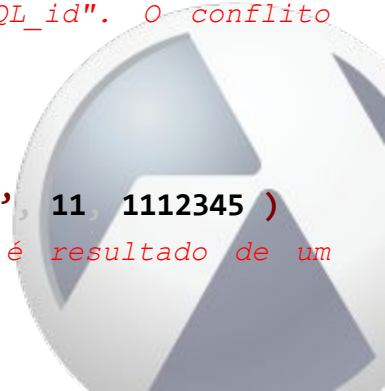
```
INSERT INTO outroexemplosql (id_2, id_exemplosql) VALUES ( 'abcd', 99999 )
```

A instrução INSERT conflitou com a restrição do FOREIGN KEY "FK_outroExemploSQL_id". O conflito ocorreu no banco de dados "DatabaseName", tabela "dbo.exemploSQL", column 'id'.

Inserção em campo calculado:

```
INSERT INTO exemplosql (texto_curto_naonulo, numero_check, calculado) VALUES ( 'abcd', 11, 1112345 )
```

A coluna "calculado" não pode ser modificada porque é uma coluna computada ou é resultado de um operador UNION.





- 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
```





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 (1)  
FROM SomeTable;
```



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



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.



- 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**.



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

```
TRUNCATE TABLE Sales.SalesPerson;
```





- **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%';
```





Erros comuns que devem ser evitados:

Os erros mais comuns em uma instrução DELETE são referentes à violação de integridade entre as tabelas.

Não se pode remover uma primary key se esta está em uso como foreign key em outra tabela.

```
INSERT INTO exemplosql(texto_curto_naonulo, numero_check)
```

```
output      inserted.* --Só por conferência
```

```
VALUES      ( 'TESTE',11 )
```

```
INSERT INTO outroexemplosql (id_2, id_exemplosql) VALUES ( 99, @@identity )
```

```
DELETE FROM exemplosql WHERE texto_curto_naonulo = 'TESTE'
```

A instrução DELETE conflitou com a restrição do REFERENCE "FK_outroExemploSQL_id". O conflito ocorreu no banco de dados "DatabaseName", tabela "dbo.outroExemploSQL", column 'id_exemploSQL'.

Pode-se inspecionar o que foi deletado utilizando-se o comando OUTPUT :

```
SELECT * FROM exemplosql WHERE texto_curto_naonulo = 'TESTE'
```

```
DELETE FROM outroexemplosql output deleted.*
```

```
WHERE id_exemplosql = ?? --ID do select anterior
```

```
DELETE FROM exemplosql output deleted.* WHERE id = ?? --ID do select anterior
```





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





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

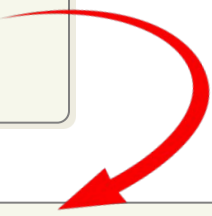
```
UPDATE SomeTable  
SET Column = Value  
WHERE SearchExpression
```

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




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



Erros comuns que devem ser evitados:

Atualizar para NULL uma coluna que não permite nulos:

```
UPDATE exemplosql SET      texto_curto_naonulo = NULL WHERE  texto_curto_naonulo = 'TESTE'
```

Não é possível inserir o valor NULL na coluna 'texto_curto_naonulo', tabela 'TesteAulaRemota.dbo.exemploSQL'; a coluna não permite nulos. Falha em UPDATE.

Exceder o tamanho de um campo: (texto_curto_naonulo é um VARCHAR(5))

```
UPDATE exemplosql SET      texto_curto_naonulo = 'abcdefg' WHERE  texto_curto_naonulo = 'TESTE'
```

Os dados de sequência ou binários estão truncados na tabela 'TesteAulaRemota.dbo.exemploSQL', coluna 'texto_curto_naonulo'. Valor truncado: 'abcde'.

Tipos incompatíveis:

```
UPDATE exemplosql SET      numero_check = 'dez' WHERE  texto_curto_naonulo = 'TESTE'
```

Falha ao converter o varchar valor 'dez' para o tipo de dados int.

Tentativa de atualização em campo auto incremental:

```
UPDATE exemplosql SET      id = 1 WHERE  texto_curto_naonulo = 'TESTE'
```

Não é possível atualizar a coluna de identidade 'id'.





Erros comuns que devem ser evitados:

Valores não passaram no teste de validação (check) : (numero_check >= 0)

```
UPDATE exemplosql SET      numero_check = -1 WHERE  texto_curto_naonulo = 'TESTE'
```

A instrução UPDATE conflitou com a restrição do CHECK "CK_exemploSQ_numer__74AE54BC". O conflito ocorreu no banco de dados "DatabaseName", tabela "dbo.exemploSQL", column 'numero_check'.

Violação de constraint de unicidade : (CONSTRAINT UQ_texto_curto_naonulo UNIQUE)

```
UPDATE exemplosql SET      texto_curto_naonulo = 'abcd' WHERE  id = 1
```

(1 linhas afetadas)

```
UPDATE exemplosql SET      texto_curto_naonulo = 'abcd' WHERE  id = 2
```

Violação da restrição UNIQUE KEY 'UQ_exemploSQL_texto_curto_naonulo'. Não é possível inserir a chave duplicada no objeto 'dbo.exemploSQL'. O valor de chave duplicada é (abcd).

Violação de constraint de chave primária na tabela dependente:

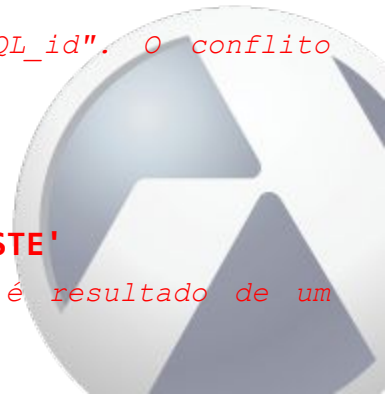
```
UPDATE outroexemplosql SET      id_exemplosql = 99999
```

A instrução UPDATE conflitou com a restrição do FOREIGN KEY "FK_outroExemploSQL_id". O conflito ocorreu no banco de dados "DatabaseName", tabela "dbo.exemploSQL", column 'id'.

Atualização em campo calculado:

```
UPDATE exemplosql SET      calculado = 1112345 WHERE  texto_curto_naonulo = 'TESTE'
```

A coluna "calculado" não pode ser modificada porque é uma coluna computada ou é resultado de um operador UNION.





Data Definition Language

CREATE TABLE Aluno

```
(  
  Matricula int not null IDENTITY (500, 1)  
  , Nome varchar(20)  
  , CONSTRAINT pkAluno  
    PRIMARY KEY (Matricula)  
);
```

Matricula	Nome
500	José
501	Pedro
502	Mario

CREATE TABLE Prova

```
(  
  idProva int NOT NULL IDENTITY (1, 1)  
  , Matricula int NOT NULL  
  , Nota decimal(4,2) NOT NULL  
  , CONSTRAINT pkProva PRIMARY KEY (idProva)  
  , CONSTRAINT fkProva FOREIGN KEY (Matricula)  
    REFERENCES Aluno(Matricula)  
);
```

idProva	Matricula	Nota
1	500	9
2	500	8
3	502	7
4	502	3
5	502	1



Data Manipulation Language Exemplos

Inserir um aluno chamado Matheus cuja nota na primeira prova seja 10

`Insert into Aluno (Nome) VALUES ('Matheus')`

`Insert into Prova (Matricula, Nota) VALUES (503, 10)`

Remover o aluno José e todas as provas por ele feitas

`Delete from prova where Matricula = 500`

`Delete from aluno where Matricula = 500`

Alterar as provas 3,4 e 5, pois esta foi feita por um novo aluno, Felipe (matricula 504)

`Insert into Aluno (Nome) VALUES ('Felipe')`

`Update prova set Matricula = 504 where matricula = 502`

Matricula	Nome
501	Pedro
502	Mario
503	Matheus
504	Felipe

idProva	Matricula	Nota
3	504	7
4	504	3
5	504	1
6	503	10



Obrigado!

Aula Gravada por:

Prof. Msc. Gustavo Bianchi Maia

gustavo.maia@faculdadeimpacta.com.br

Material criado e oferecido por :

Prof. Sand Jacques Onofre

Sand.onofre@faculdadeimpacta.com.br

