

# Comandos DML e DQL

Prof. Têlvio Orrú  
telvio.orrú@docente.unip.br

Universidade Paulista UNIP  
BD@2023

**Aula 3**

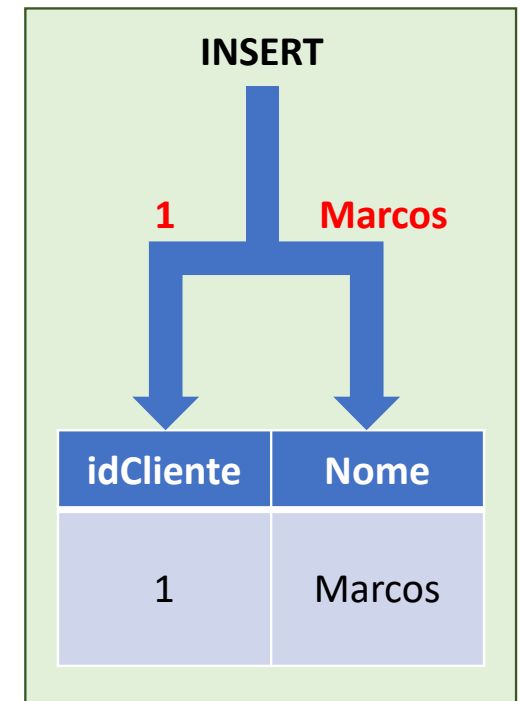
# Composição dos Comandos DML

DML - Data Manipulation Language:

Command	Description
INSERT	Creates a record
UPDATE	Modifies records
DELETE	Deletes records

# Sobre o INSERT

- ❑ Sua principal função é permitir a **inclusão de dados (registros) nas tabelas** do banco;
- ❑ Sempre que utilizado **modifica o conteúdo da tabela** para o qual foi aplicado;
- ❑ Os dados inseridos devem ser **compatíveis com as colunas e restrições** da tabela (esquema).



# Sintaxe do Comando INSERT

## HOW-TO TUTORIAL

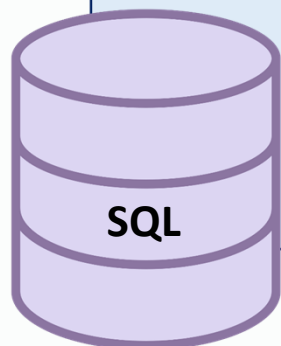
```
INSERT INTO <nm-tab> [(<nm-col> [,nm-col])] VALUES (<valores>)
```

Onde:

**nm-tab:** nome da tabela que será incluído o registro

**nm-col:** nome das colunas que receberão os valores da operação

**valores:** conteúdo que será atribuído às colunas



# Exemplo 1 de Utilização

## (Especificando o Nome dos Campos)

```
CREATE TABLE Funcionario(  
    idFunc INT NOT NULL PRIMARY KEY,  
    nome VARCHAR(50) NULL,  
    endereco VARCHAR(100) NULL,  
    cidade VARCHAR(50) NULL,  
    estado CHAR(2) NULL,  
    email VARCHAR(50) NULL,  
    dataNascto DATE NULL  
)
```

I  
N  
S  
E  
R  
Ç  
Ã  
O

idFunc	nome	endereco	cidade	estado	email	dataNascto
1	Nathan Cirillo	Av. Humberto	Jundiaí	SP	email	1988-08-17

C  
O  
N  
S  
U  
L  
T  
A

```
INSERT INTO Funcionario (idFunc,nome,endereco,cidade,estado,email,dataNascto)  
VALUES (1,'Nathan Cirillo','Av. Humberto','Jundiaí','SP','email','17-08-1988')
```

# Exemplo 2 de Utilização

(Omitindo o Nome dos Campos na Sequência)

```
CREATE TABLE Funcionario(  
  idFunc INT NOT NULL PRIMARY KEY,  
  nome VARCHAR(50) NULL,  
  endereco VARCHAR(100) NULL,  
  cidade VARCHAR(50) NULL,  
  estado CHAR(2) NULL,  
  email VARCHAR(50) NULL,  
  dataNascto DATE NULL  
)
```

I  
N  
S  
E  
R  
Ç  
Ã  
O

idFunc	nome	endereco	cidade	estado	email	dataNascto
1	Nathan Cirilo	Av. Humberto	Jundiaí	SP	email	1988-08-17
2	Luciana Prado	Ferrovários	Jundiaí	SP	mail2	1989-04-20

C  
O  
N  
S  
U  
L  
T  
A

INSERT INTO Funcionario VALUES  
(2,'Luciana Prado','Ferrovários','Jundiaí','SP','mail2','20-04-1989')

# Exemplo 3 de Utilização

## (Inserindo Múltiplos Registros)

```
CREATE TABLE Funcionario(  
  idFunc INT NOT NULL PRIMARY KEY,  
  nome VARCHAR(50) NULL,  
  endereco VARCHAR(100) NULL,  
  cidade VARCHAR(50) NULL,  
  estado CHAR(2) NULL,  
  email VARCHAR(50) NULL,  
  dataNascto DATE NULL  
)
```

I  
N  
S  
E  
R  
Ç  
Ã  
O

INSERT INTO Funcionario VALUES

(3,'João da Silva','Imigrantes','Varzea','SP','mail3','04/08/1998'),  
(4,'Fernando Ap','Rua Lima','Jundiaí','SP','mail4','05/07/1982'),  
(5,'Petrine','Rua Cica','Jundiaí','SP','mail5','07/09/1964')

idFunc	nome	endereco	cidade	estado	email	dataNascto
1	Nathan Cirilo	Av. Humberto	Jundiaí	SP	email	1988-08-17
2	Luciana Prado	Ferrovários	Jundiaí	SP	mail2	1989-04-20
3	João da Silva	Imigrantes	Varzea	SP	mail3	1998-08-04
4	Fernando Ap	Rua Lima	Jundiaí	SP	mail4	1982-07-05
5	Petrine	Rua Cica	Jundiaí	SP	mail5	1964-09-07

C  
O  
N  
S  
U  
L  
T  
A

# Exemplo 4 de Utilização

## (Inserindo Apenas os Campos Obrigatórios)

```
CREATE TABLE Funcionario(  
  idFunc INT NOT NULL PRIMARY KEY,  
  nome VARCHAR(50) NULL,  
  endereco VARCHAR(100) NULL,  
  cidade VARCHAR(50) NULL,  
  estado CHAR(2) NULL,  
  email VARCHAR(50) NULL,  
  dataNascto DATE NULL  
)
```

INSERÇÃO

idFunc	nome	endereco	cidade	estado	email	dataNascto
1	Nathan Cirillo	Av. Humberto	Jundiaí	SP	email	1988-08-17
2	Luciana Prado	Ferrovários	Jundiaí	SP	mail2	1989-04-20
3	João da Silva	Imigrantes	Varzea	SP	mail3	1998-08-04
4	Fernando Ap	Rua Lima	Jundiaí	SP	mail4	1982-07-05
5	Petrine	Rua Cica	Jundiaí	SP	mail5	1964-09-07
6	NULL	NULL	NULL	NULL	NULL	NULL
7	NULL	NULL	NULL	NULL	NULL	NULL
8	NULL	NULL	NULL	NULL	NULL	NULL
9	NULL	NULL	NULL	NULL	NULL	NULL
10	NULL	NULL	NULL	NULL	NULL	NULL

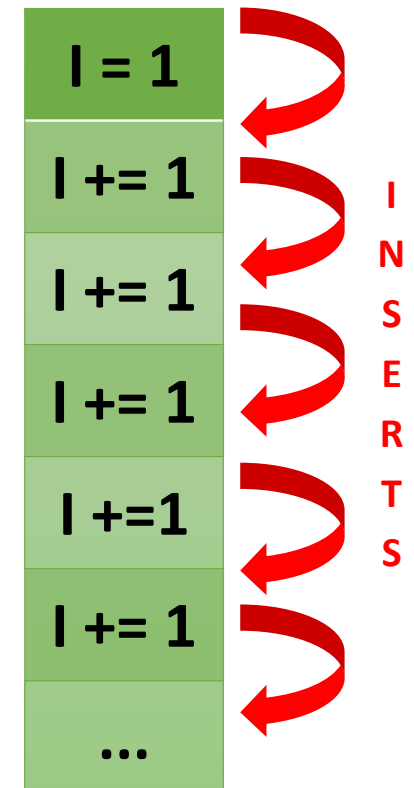
CONSULTA

```
INSERT INTO Funcionario (idFunc) VALUES (6),(7),(8),(9),(10)
```



# Campo de Auto Numeração (*Identity*)

- ❑ A propriedade *identity* é usada para definir um campo como **auto incremento**;
- ❑ **Números sequenciais** são criados automaticamente para os registros inseridos;
- ❑ Cada tabela pode conter **somente um campo** como auto incremento (*identity*);
- ❑ Salienta-se que os seus valores **não podem ser inseridos ou modificados**.



# Sintaxe do Comando **IDENTITY**

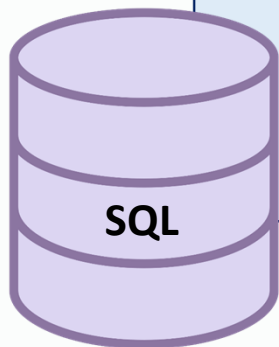
HOW-TO TUTORIAL

**IDENTITY** [**<início>**,**<incremento>**]

Onde:

**Início:** valor usado para a primeira linha carregada


**Incremento:** valor que será adicionado ao valor da linha anterior



# Exemplo de Utilização

```
CREATE TABLE Cargo(  
    idCargo INT IDENTITY PRIMARY KEY,  
    sigla CHAR(2) NOT NULL,  
    nome VARCHAR(30) NULL UNIQUE  
)
```

```
INSERT INTO Cargo (sigla, nome) VALUES  
(‘AI’, ‘Auxiliar de Informática’),  
(‘PC’, ‘Programador de Computador’),  
(‘TI’, ‘Técnico de Informática’),  
(‘AN’, ‘Analista’)
```

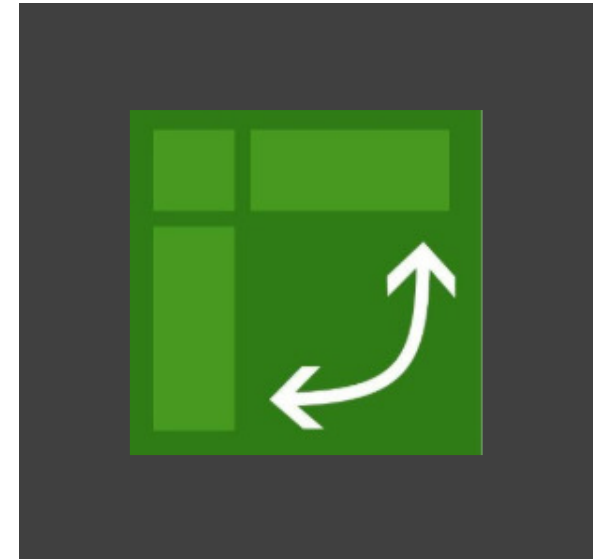


The diagram illustrates the process of inserting data into a table and then querying it. A red arrow labeled 'INSERTÃO' points from the INSERT statement to the table. Another red arrow labeled 'CONSULTA' points from the table to the right, indicating a query operation. The table itself has four rows, with the first row circled in red.

idCargo	sigla	nome
1	AI	Auxiliar de Informática
2	PC	Programador de Computador
3	TI	Técnico de Informática
4	AN	Analista

# Campo Calculado

- ❑ São **colunas virtuais** que *não estão armazenadas fisicamente na tabela de dados*;
- ❑ Elas são **calculadas com base nos valores de outros campos** do próprio registro;
- ❑ A sua utilização lista o resultado de operações **sem onerar o armazenamento físico**.



# Exemplo de Utilização

```
CREATE TABLE Impostos(  
  val REAL NOT NULL,  
  tx REAL NOT NULL,  
  tot AS (val * tx)  
)
```

**Campo Calculado**  
*definido com base em  
outros campos*

	<i>val</i>	<i>tx</i>	<i>tot</i>
1	100	0,1	10
2	550	0,2	110

```
INSERT INTO Impostos VALUES (100,0.1), (550,0.2)
```

```
SELECT * FROM Impostos
```



# Sobre o UPDATE

- ❑ *Imagine que ao inserir dados na tabela Funcionários algum conteúdo ficou errado ou desatualizado;*
- ❑ O comando Update serve justamente **alterar esse conteúdo** sem precisar excluir a linha;
- ❑ Portanto, ele permite **atualizar um ou vários registros de uma tabela** do banco de dados;
- ❑ Os novos dados deverão ser **compatíveis com os tipos das colunas e restrições** da tabela.



# Sintaxe do Comando UPDATE

## HOW-TO TUTORIAL

```
UPDATE <nm-tab> SET  
<nm-col> = <novo_valor> [, <nm-col> = <novo_valor> ]  
WHERE <condição>
```

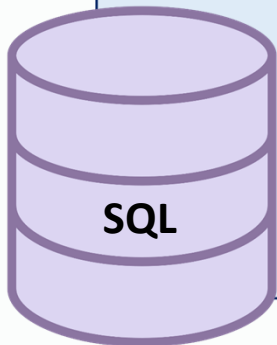
Onde:

**nm-tab:** nome da tabela que terá o conteúdo alterado

**nm-col:** nome da coluna que terá o conteúdo alterado

**novo\_valor:** novo conteúdo que será atribuído ao campo

**condição:** permite especificar qual registro será alterado



# Exemplos de Utilização

```
UPDATE Funcionario SET  
nome = 'Gustavo Lanza', cidade = 'Varzea'  
WHERE idFunc=5;
```

```
UPDATE Funcionario SET  
nome = 'Fernando Aparecido'  
WHERE nome = 'Fernando Ap';
```

E se tirarmos  
o WHERE?

idFunc	nome	endereço	cidade	estado	email	dataNascto
1	Nathan Cirillo	Av. Humberto	Jundiaí	SP	email	1988-08-17
2	Luciana Prado	Ferrovários	Jundiaí	SP	mail2	1989-04-20
3	João da Silva	Imigrantes	Varzea	SP	mail3	1998-08-04
4	Fernando Aparecido	Rua Lima	Jundiaí	SP	mail4	1982-07-05
5	Gustavo Lanza	Rua Cica	Varzea	SP	mail5	1964-09-07
6	NULL	NULL	NULL	NULL	NULL	NULL
7	NULL	NULL	NULL	NULL	NULL	NULL
8	NULL	NULL	NULL	NULL	NULL	NULL
9	NULL	NULL	NULL	NULL	NULL	NULL
10	NULL	NULL	NULL	NULL	NULL	NULL



# Update sem WHERE

```
UPDATE Funcionario SET  
cidade = 'Jundiaí';
```

Sem **WHERE <condição>** todos os registros serão modificados!

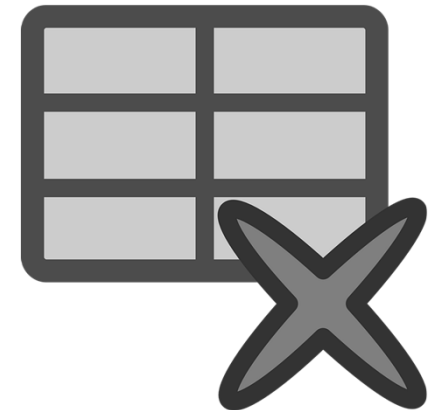
idFunc	nome	endereco	cidade	estado	email	dataNascto
1	Nathan Cirilo	Av. Humberto	Jundiaí	SP	email	1988-08-17
2	Luciana Prado	Ferrovários	Jundiaí	SP	mail2	1989-04-20
3	João da Silva	Imigrantes	Jundiaí	SP	mail3	1998-08-04
4	Fernando Aparecido	Rua Lima	Jundiaí	SP	mail4	1982-07-05
5	Gustavo Lanza	Rua Cica	Jundiaí	SP	mail5	1964-09-07
6	NULL	NULL	Jundiaí	NULL	NULL	NULL
7	NULL	NULL	Jundiaí	NULL	NULL	NULL
8	NULL	NULL	Jundiaí	NULL	NULL	NULL
9	NULL	NULL	Jundiaí	NULL	NULL	NULL
10	NULL	NULL	Jundiaí	NULL	NULL	NULL



Use com atenção!

# Sobre o DELETE

- ☐ Serve para **remover um ou mais registros** existentes em uma determinada tabela;
- ☐ O **registro inteiro é apagado**, não permitindo excluir apenas o valor individual da coluna;
- ☐ É **executado sem qualquer confirmação** de exclusão, portanto utilize com cuidado.



# Sintaxe do Comando **DELETE**

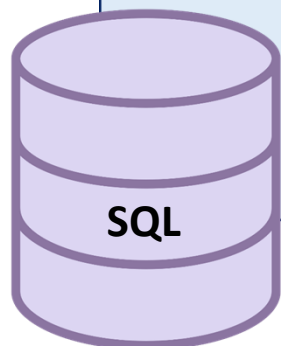
## HOW-TO TUTORIAL

**DELETE FROM** <nm-tab>  
**WHERE** <condição>

**Onde:**

**nm-tab:** nome da tabela que terá o(s) registro(s) excluído(s).

**condição:** permite especificar qual(is) registro(s) será(ao) apagado(s).



# Exemplos de Utilização

DELETE FROM Funcionario  
WHERE idFunc=9 OR idFunc=10;

idFunc	nome	endereco	cidade	estado	email	dataNasc
1	Nathan Cirilo	Av. Humberto	Jundiaí	SP	email	1988-08-17
2	Luciana Prado	Ferrovários	Jundiaí	SP	mail2	1989-04-20
3	João da Silva	Imigrantes	Jundiaí	SP	mail3	1998-08-04
4	Fernando Aparecido	Rua Lima	Jundiaí	SP	mail4	1982-07-05
5	Gustavo Lanza	Rua Cica	Jundiaí	SP	mail5	1964-09-07
6	NULL	NULL	Jundiaí	NULL	NULL	NULL
7	NULL	NULL	Jundiaí	NULL	NULL	NULL
8	NULL	NULL	Jundiaí	NULL	NULL	NULL
9	NULL	NULL	Jundiaí	NULL	NULL	NULL
10	NULL	NULL	Jundiaí	NULL	NULL	NULL

DELETE FROM Funcionario  
WHERE nome IS NULL;

idFunc	nome	endereco	cidade	estado	email	dataNasc
1	Nathan Cirilo	Av. Humberto	Jundiaí	SP	email	1988-08-17
2	Luciana Prado	Ferrovários	Jundiaí	SP	mail2	1989-04-20
3	João da Silva	Imigrantes	Jundiaí	SP	mail3	1998-08-04
4	Fernando Aparecido	Rua Lima	Jundiaí	SP	mail4	1982-07-05
5	Gustavo Lanza	Rua Cica	Jundiaí	SP	mail5	1964-09-07
6	NULL	NULL	Jundiaí	NULL	NULL	NULL
7	NULL	NULL	Jundiaí	NULL	NULL	NULL
8	NULL	NULL	Jundiaí	NULL	NULL	NULL

# Delete sem Where

- ❑ Utilize o comando delete com bastante atenção e **não se esqueça de usar o WHERE <condição>**;
- ❑ Caso o WHERE <condição> não for informado, **todos os registros serão apagados**;
- ❑ Conforme já mencionado, **nenhuma confirmação será solicitada**.

Sem **WHERE <condição>**  
*todos os registros serão  
apagados!*

**DELETE FROM** Funcionario;

# Sobre o TRUNCATE

- ❑ O comando Truncate é uma **variação especial do Delete** apresentado anteriormente;
- ❑ Diferente do Delete, o Truncate sempre irá **limpar a tabela por completo** (não há WHERE);
- ❑ Além disso, ele **reseta os campos do tipo *identity*** para os seus respectivos valores iniciais;
- ❑ Por não gerar tanto log (um para cada registro), ele acaba sendo **mais rápido do que o Delete**.



# Delete vs Truncate

idFunc	nome	endereco	cidade	estado	email	dataNascto
1	Nathan Cirillo	Av. Humberto	Jundiaí	SP	email	1988-08-17
2	Luciana Prado	Ferrovários	Jundiaí	SP	mail2	1989-04-20
3	João da Silva	Imigrantes	Jundiaí	SP	mail3	1998-08-04
4	Fernando Aparecido	Rua Lima	Jundiaí	SP	mail4	1982-07-05
5	Gustavo Lanza	Rua Cica	Jundiaí	SP	mail5	1964-09-07

**DELETE FROM**

**TRUNCATE TABLE**

idFunc	nome	endereco	cidade	estado	email	dataNascto
6	pedro	rua	jundiaí	sp	mail	1990-08-28

idFunc	nome	endereco	cidade	estado	email	dataNascto
1	pedro	rua	jundiaí	sp	mail	1990-08-28

# Composição dos Comandos DQL

## DQL - Data Query Language:

Command	Description
SELECT	Retrieves certain records from one or more tables



# Sobre o SELECT

- ☐ O comando **SELECT** é *mais utilizado do que qualquer outra declaração SQL existente*;
- ☐ O seu principal objetivo é **permitir a consulta de dados**, retornando um **conjunto de registros**;



- ☐ Faz a apresentação dos dados, porém o **conteúdo das tabelas consultadas não é alterado**;
- ☐ **Critérios de busca poderão ser estabelecidos** para filtrar melhor os dados requisitados.

# Sintaxe do Comando **SELECT**

```
SELECT [DISTINCT | ALL] <nm-col> [, <nm-col>]  
FROM <nm-tab> [, <nm-tab>]  
WHERE <condicao>  
GROUP BY <nm-col-agp>  
HAVING <condicao-agp>  
ORDER BY <nm-col-ord> [ASC | DESC]
```

**DISTINCT** ou **ALL**: elimina duplicidades nos registros ou não (*padrão*).

**nm-col**: nome da(s) coluna(s) que será(ão) consultada(s).

**nm-tab**: nome da(s) tabela(s) que será(ão) consultada(s).

**condicao**: critério de busca utilizado para a seleção dos registros.

**nm-col-agp**: especifica o(s) campo(s) que será(ão) agrupado(s).

**condicao-agp**: critério para determinar o agrupamento dos dados.

**nm-col-ord**: ordena o resultado das consultas pelo campo definido.



# Exemplo 1 de Utilização

(Buscando Todos os Dados)

Selecione **TODOS OS DADOS** da tabela!

```
SELECT * FROM Funcionario;
```

Todas as colunas da tabela!

idFunc	nome	endereço	cidade	estado	email	dataNascto
1	Nathan Cirillo	Av. Humberto	Jundiaí	SP	email	1988-08-17
2	Luciana Prado	Ferrovários	Jundiaí	SP	mail2	1989-04-20
3	João da Silva	Imigrantes	Jundiaí	SP	mail3	1998-08-04
4	Fernando Aparecido	Rua Lima	Jundiaí	SP	mail4	1982-07-05
5	Gustavo Lanza	Rua Cica	Jundiaí	SP	mail5	1964-09-07



# SELECT \* FROM TABLE

1. Unnecessary I/O
2. Increased Network traffic
3. More Application memory
4. Depends on Column Order
5. Fragile Views
6. Conflict in JOIN Query
7. Risky while copying data

## Exemplo 2 de Utilização

(Somente Campos Necessários)

Informe os **Campos Desejados!**

```
SELECT nome, endereco FROM Funcionario;
```

	nome	endereco
1	Nathan Cirillo	Av. Humberto
2	Luciana Prado	Ferrovários
3	João da Silva	Imigrantes
4	Fernando Aparecido	Rua Lima
5	Gustavo Lanza	Rua Cica

Apresenta **somente os campos escolhidos!**

# Exemplo 3 de Utilização

(Utilizando Critério de Busca)

*Observe a importância de conhecer os operadores. Eles possibilitam o refinamento da consulta!*



Critério a ser atendido  
(buscado).

```
SELECT * FROM Funcionario WHERE dataNascto='1998-08-04';
```

	idFunc	nome	endereço	cidade	estado	email	dataNascto
1	3	João da Silva	Imigrantes	Jundiaí	SP	mail3	1998-08-04

# Outros Exemplos de Consultas

```
SELECT nome FROM Funcionario  
ORDER BY nome;
```



	nome
1	Fernando Aparecido
2	Gustavo Lanza
3	João da Silva
4	Luciana Prado
5	Nathan Cirilo

```
SELECT nome FROM Funcionario  
ORDER BY nome DESC;
```



	nome
1	Nathan Cirilo
2	Luciana Prado
3	João da Silva
4	Gustavo Lanza
5	Fernando Aparecido

```
SELECT DISTINCT cidade  
FROM Funcionario;
```



	cidade
1	Jundiaí

# Dúvidas?

telvio.orr@docente.unip.br





Mãos à Obra!!!



Envie por e-mail conforme o padrão apresentado na Aula 1