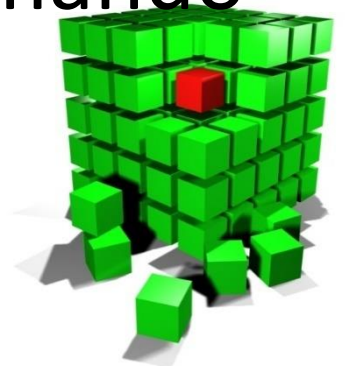
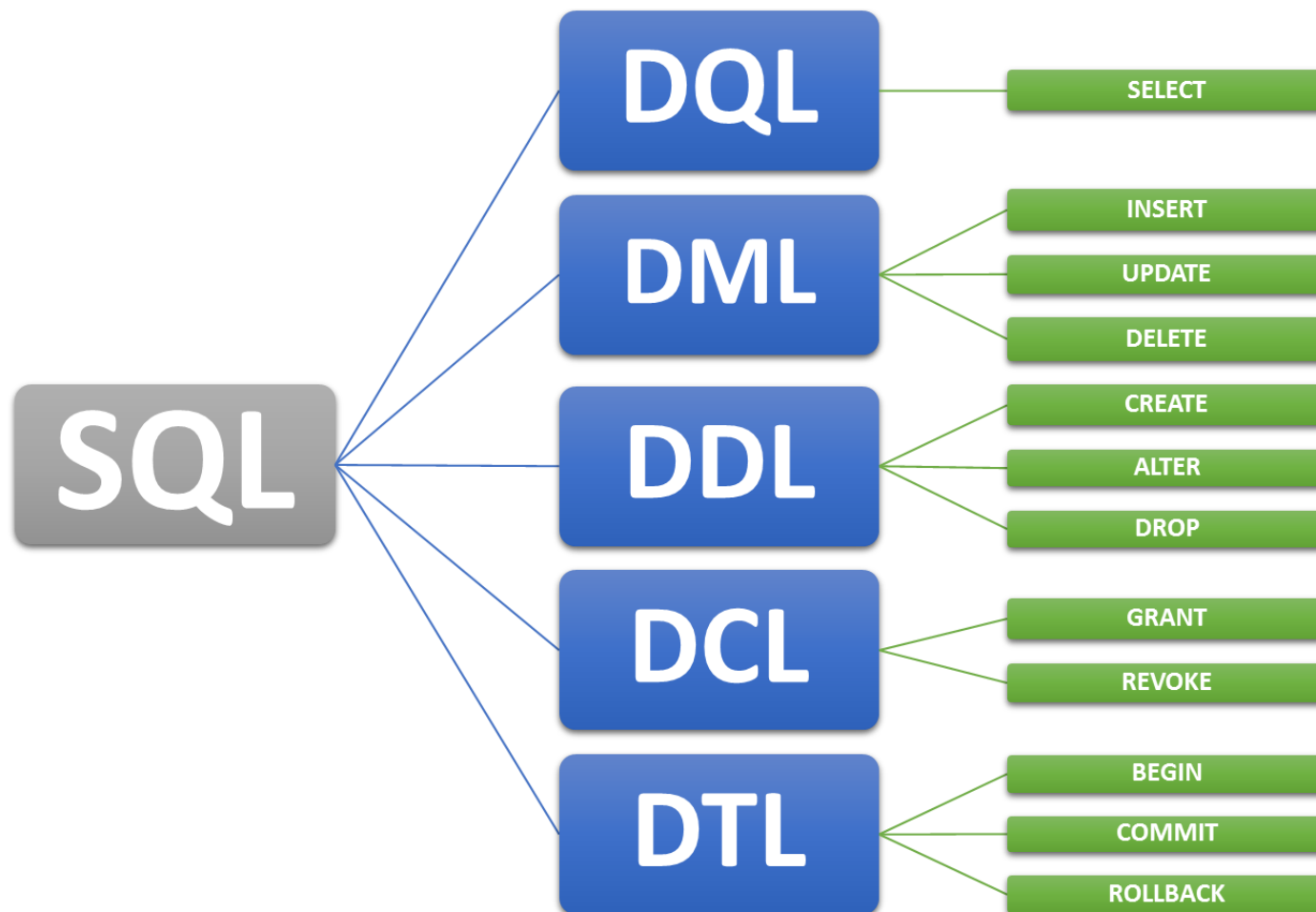


- **Aula 009 – Linguagem SQL**
  - Comandos básicos da linguagem SQL.
  - Criação de banco de dados e tabelas.
  - Inserção de dados.
  - Consultas básicas utilizando o comando SELECT e a cláusula WHERE.
  - Cláusula ORDER BY.



## ■ Categorias de comandos



## ■ Comandos de definição de dados

Comando ou Opção	Descrição
CREATE SCHEMA AUTHORIZATHION	Cria um esquema
CREATE TABLE	Cria uma nova tabela
NOT NULL	Não permite valores nulos
UNIQUE	Não permite valores duplicados
PRIMARY KEY	Define a chave primária da tabela
FOREIGN KEY	Define a chave estrangeira
DEFAULT	Define um valor padrão
CHECK	Valida os dados de um atributo
CREATE INDEX	Cria um índice para uma tabela
CREATE VIEW	Cria uma visão
ALTER TABLE	Modifica a estrutura de uma tabela



## ■ Comandos de definição de dados

Comando ou Opção	Descrição
CREATE TABLE AS	Cria uma nova tabela, baseada em uma consulta no esquema de banco de dados dos usuário
DROP TABLE	Exclui uma tabela e todos os seus dados
DROP INDEX	Exclui um índice
DROP VIEW	Exclui uma visão



## ■ Comandos de manipulação de dados

Comando ou Opção	Descrição
INSERT	Insere linhas em uma tabela
SELECT	Seleciona atributos de uma ou mais tabelas
WHERE	Filtra a seleção de linhas
GROUP BY	Agrupar as linhas selecionadas
HAVING	Restringe a seleção de linhas agrupadas
ORDER BY	Ordena as linhas selecionadas
UPDATE	Modifica os valores de um atributo em uma ou mais tabelas
DELETE	Exclui linhas de uma tabela
COMMIT	Salva as alterações de forma permanente
ROLLBACK	Restaura os dados para seus valores originais

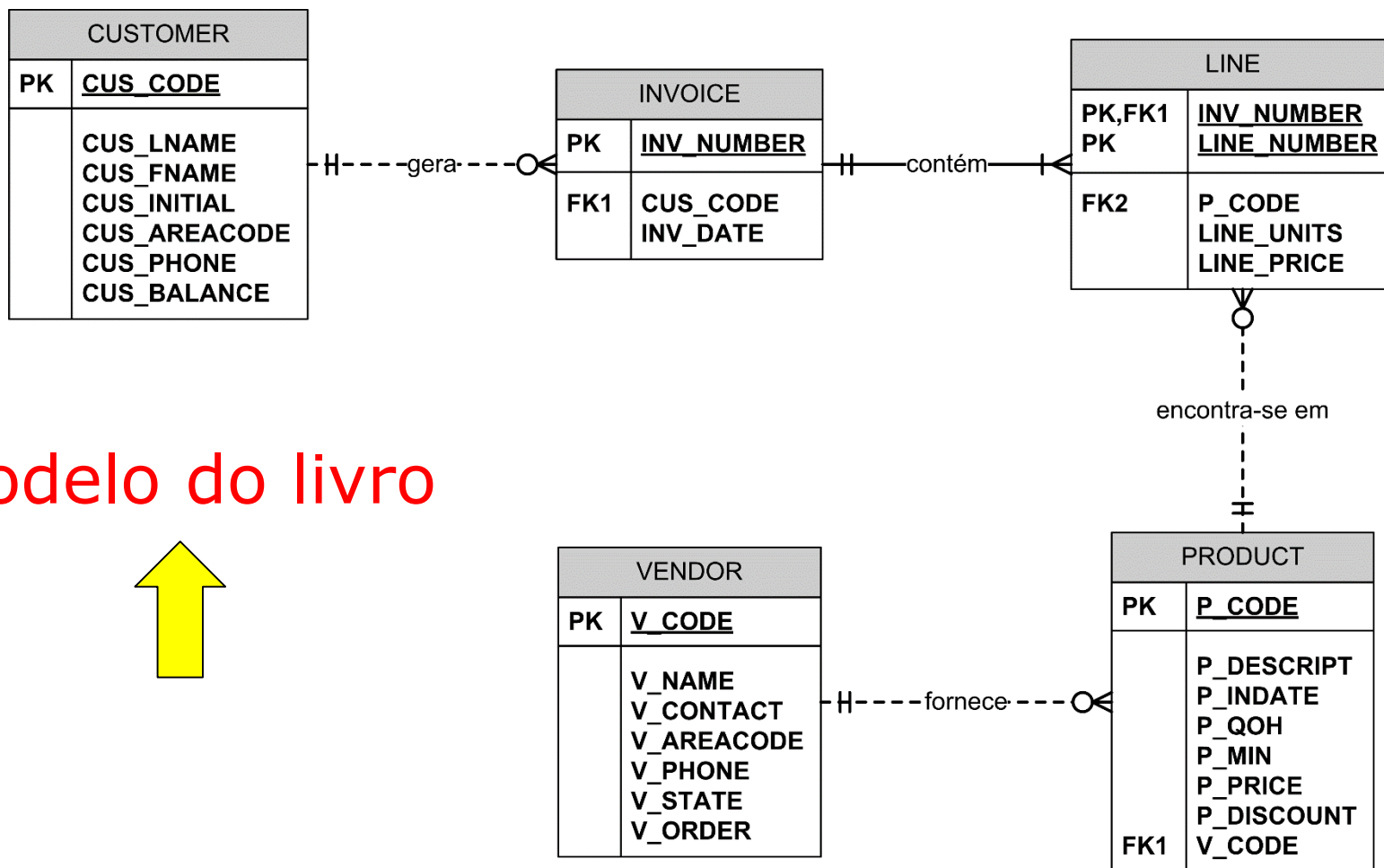


## ■ Comandos de manipulação de dados

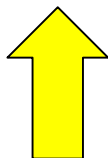
Comando ou Opção	Descrição
=, <, >, <=, >=, <>, !=	Operadores de comparação
AND, OR, NOT	Operadores lógicos
BETWEEN	Verifica se o valor do atributo está dentro de uma determinada faixa
IS NULL	Verifica se o valor do atributo é nulo
LIKE	Verifica se o valor do atributo coincide com determinado padrão de caracteres
IN	Verifica se o valor do atributo coincide com qualquer valor dentro de uma lista
EXISTS	Verifica se uma subconsulta retorna uma linha
DISTINCT	Limita os valores a valores exclusivos



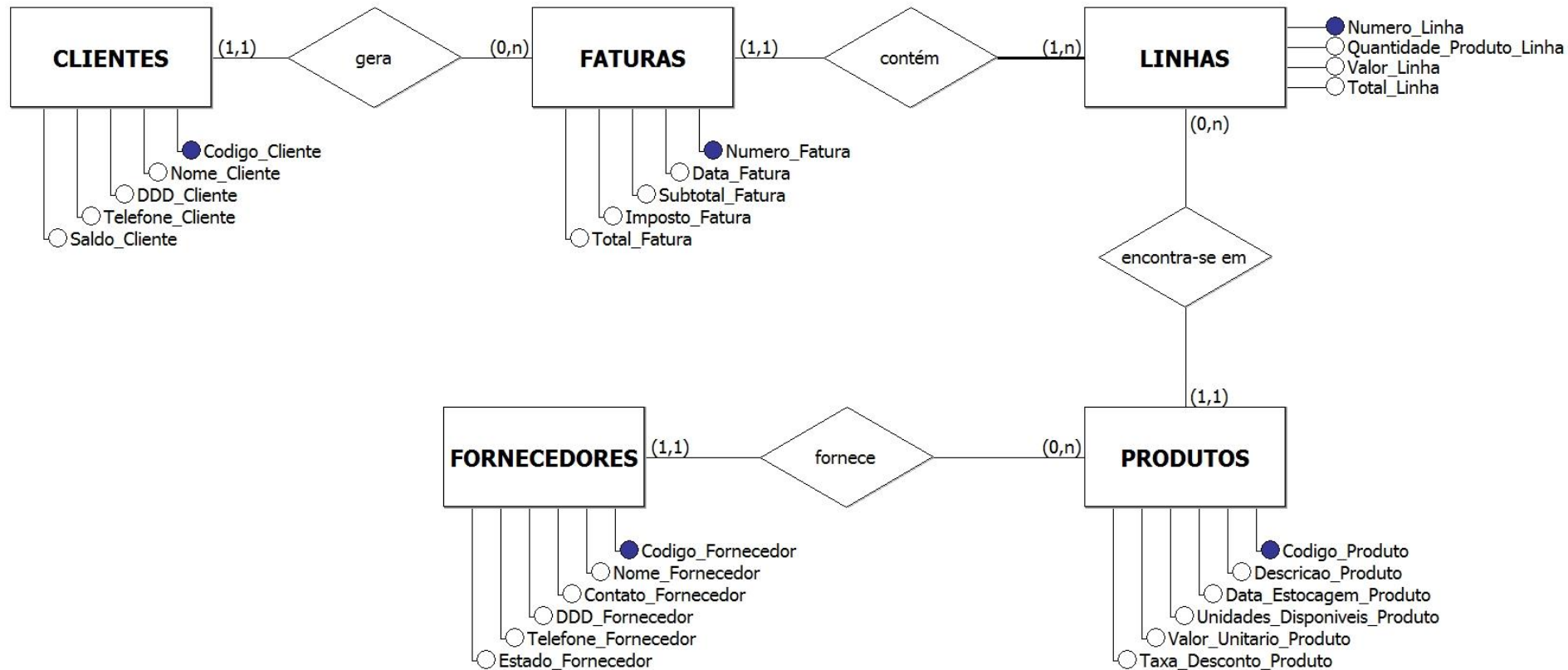
## ■ Modelo de banco de dados – Visio



Modelo do livro

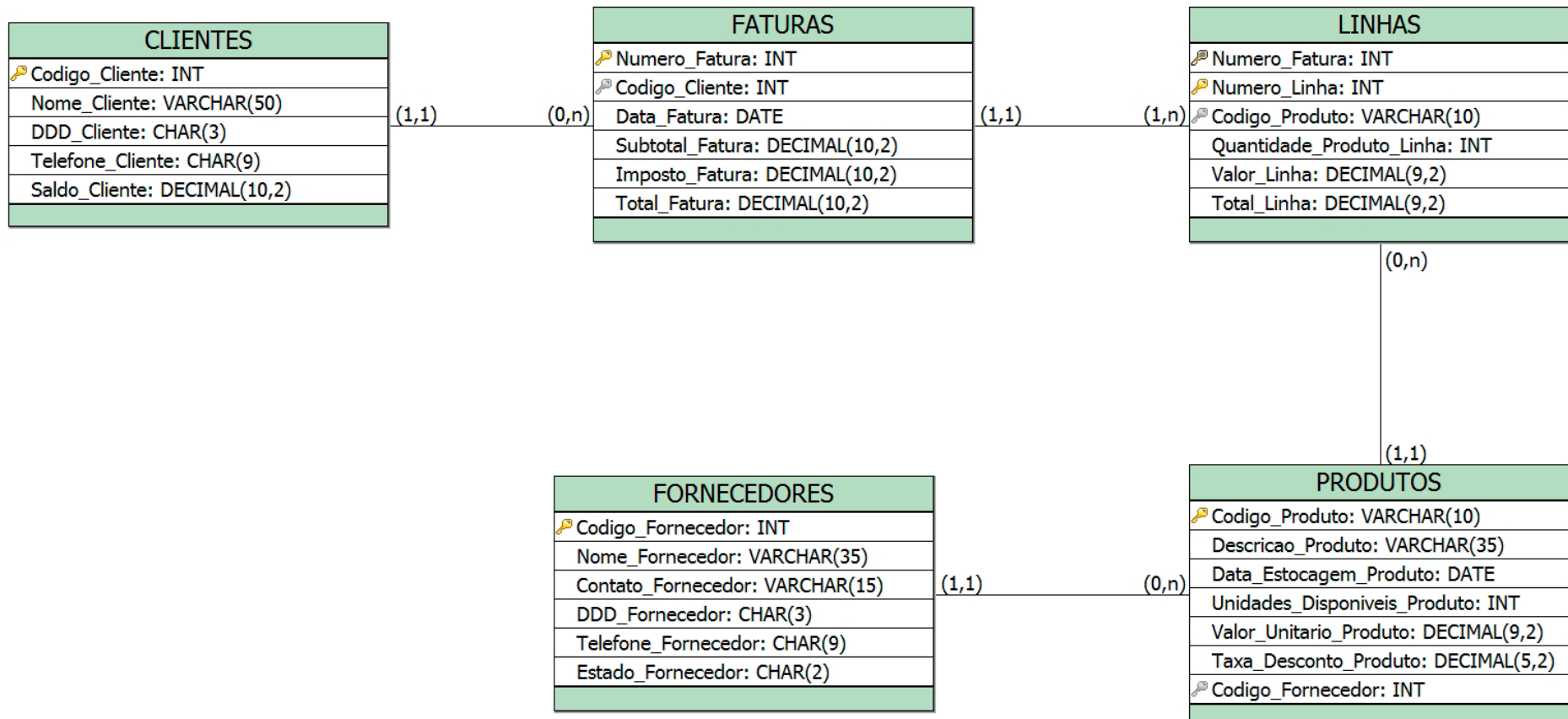


## ■ Modelo conceitual adaptado





## ■ Modelo lógico adaptado



## ■ Apresentação dos comandos SQL



No caso, iremos utilizar um modelo simples para conhecer os comandos e complementaremos o aprendizado por meio de listas de exercícios.

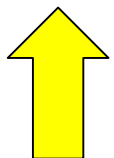


## ■ Comando **CREATE DATABASE**

- O comando **CREATE DATABASE** é utilizado para criar um banco de dados. Sua sintaxe básica é dada por:

**CREATE DATABASE** [Nome do Banco]

**GO**



Utilizado pelo SQL Server



## ■ Comando **USE DATABASE**

- O comando **USE DATABASE** é utilizado para habilitar o contexto de um banco de dados. Sua sintaxe básica é dada por:

**USE DATABASE** [Nome do Banco]


**GO**



## ■ Comando **CREATE TABLE**

- O comando **CREATE TABLE** é utilizado para criar uma tabela. Sua sintaxe básica é dada por:

```
CREATE TABLE [Nome da Tabela] (  
    Campo_1 Tipo_de_Dados [Restrições]  
)  
GO
```



Pode ser uma PK, FK, índices, etc



- Criando um banco de dados em T-SQL

-- Cria o banco de dados

```
CREATE DATABASE Aula09
```

```
GO
```

-- Habilita o contexto

```
USE Aula09
```

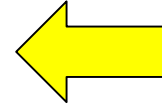
```
GO
```



## ■ Criando um banco de dados em T-SQL

-- Cria a tabela FUNCIONARIOS

```
CREATE TABLE FUNCIONARIOS (  
    ID            INT    PRIMARY KEY,  
    Nome          VARCHAR(25),  
    Sexo          CHAR(1),  
    Admissao      DATE,  
    Salario       DECIMAL(10,2)  
)
```



GO



## ■ Comando **SET DATEFORMAT**

- O comando **SET DATEFORMAT** é utilizado para alterar a ordem da entrada de valores referentes a datas (dia, mês e ano). Sua sintaxe básica é dada por:

```
SET DATEFORMAT [Ordem desejada]  
GO
```





## ■ Comando **SET DATEFORMAT**

-- Altera o formato de data do SQL Server

**SET DATEFORMAT DMY**

**GO**



D – Day  
M – Month  
Y – Year



## ■ Comando **INSERT INTO**

- O comando **INSERT INTO** é utilizado para inserir dados nas tabelas. Sua sintaxe básica é dada por:

```
INSERT INTO [Nome da Tabela]  
    (Coluna 1, Coluna 2, ...)
```

```
VALUES
```

```
(Valor 1, Valor 2, ...)
```



## ■ Comando **INSERT INTO**

-- Insere dados na tabela FUNCIONARIOS

```
INSERT INTO FUNCIONARIOS (  
    ID,  
    Nome,  
    Sexo,  
    Admissao,  
    Salario)  
VALUES (1, 'Maria da Silva', 'F', '10/01/2018',  
2500.00)  
GO
```



## ■ Comando **INSERT INTO**

- Se os dados já estiverem em ordem, pode-se omitir os valores de **Coluna 1**, **Coluna 2**, etc.:

```
INSERT INTO [Nome da Tabela]  
VALUES
```

```
(Valor 1, Valor 2, ...)
```



## ■ Comando **INSERT INTO**

-- Insere dados na tabela de FUNCIONARIOS

```
INSERT INTO FUNCIONARIOS VALUES (2,  
'Pedro Pereira', 'M', '25/05/2015',  
990.00)
```

```
GO
```



## ■ Comando **INSERT INTO**

- Para inserir vários registros ao mesmo tempo podemos utilizar duas sintaxes:

```
INSERT INTO Tabela VALUES (C1, C2, ...)
```

```
INSERT INTO Tabela VALUES (C1, C2, ...)
```

```
INSERT INTO Tabela VALUES (C1, C2, ...)
```

```
INSERT INTO Tabela VALUES (C1, C2, ...)
```

```
GO
```



## ■ Comando **INSERT INTO**

-- Insere dois novos registros - Sintaxe 1

```
INSERT INTO FUNCIONARIOS VALUES (3,  
'Maria Cristina', 'F', '10/09/2015',  
1200.00)
```

```
INSERT INTO FUNCIONARIOS VALUES (4,  
'Antônio Carlos', 'M', '15/05/2015',  
990.00)
```

```
GO
```



## ■ Comando **INSERT INTO**

- Para inserir vários registros ao mesmo tempo podemos utilizar duas sintaxes:

**INSERT INTO** Tabela **VALUES**

(C1, C2, ...),

(C1, C2, ...),

(C1, C2, ...) ← Aqui não tem vírgula

**GO**





## ■ Comando **INSERT INTO**

-- Insere quatro novos registros - Sintaxe 2

**INSERT INTO** FUNCIONARIOS **VALUES**

(5, 'Marcelo Augusto', 'M', '09/12/2017',  
1900.00),

(6, 'Pedro Silva', 'M', '15/11/2015',  
1050.00),

(7, 'Mônica da Silva', 'F', '12/10/2014',  
3000.00),

(8, 'Tiago Lima', 'M', '10/05/2016', 1350.50)

**GO**



## ■ Comando **SELECT... FROM**

- O comando **SELECT** é utilizado para retornar registros e informações de uma tabela:

```
SELECT      Campo1  AS  'Apelido',  
            Campo2  AS  'Apelido',  
            ...  
FROM [Nome da Tabela]
```



## ■ Comando **SELECT... FROM**

- Exibe algumas informações dos funcionários,
- utilizando aliases para algumas colunas

```
SELECT  ID AS 'Código do Funcionário',  
        Nome,  
        Sexo,  
        Salario AS 'Salário' ←  
  
FROM  FUNCIONARIOS  
  
GO
```



## ■ Comando **SELECT... FROM**

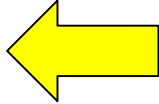
Código do Funcionário	Nome	Sexo	Salário
1	Maria da Silva	F	2500.00
2	Pedro Pereira	M	990.00
3	Maria Cristina	F	1200.00
4	Antônio Carlos	M	990.00
5	Marcelo Augusto	M	1900.00
6	Pedro Silva	M	1050.00
7	Mônica da Silva	F	3000.00
8	Tiago Lima	M	1350.50

(8 row(s) affected)



- Comando **SELECT... FROM**

- Podemos utilizar o \* como um atalho para todos os campos da tabela:

**SELECT** \*   
**FROM** [Nome da Tabela]

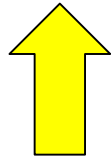


- Comando **SELECT... FROM**

-- Exibe todos os dados da  
-- tabela FUNCIONARIOS

**SELECT** \* **FROM** FUNCIONARIOS

**GO**



- Comando **SELECT... FROM**

- Para limitarmos o número de registros que serão retornados utilizamos o parâmetro **TOP**:

```
SELECT TOP 3 *  
FROM [Nome da Tabela]
```

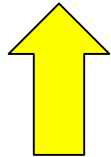


## ■ Comando **SELECT... FROM**

- Exibe todos os dados da tabela
- FUNCIONARIOS. Utiliza TOP para listar
- somente os 3 primeiros registros

```
SELECT TOP 3 * FROM FUNCIONARIOS
```

```
GO
```

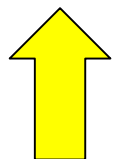




## ■ Comando **SELECT... FROM**

ID	Nome	Sexo	Salario
1	Maria da Silva	F	2500.00
2	Pedro Pereira	M	990.00
3	Maria Cristina	F	1200.00

(3 row(s) affected)



## ■ Comando **SELECT... FROM**

ID	Nome	Sexo	Salario
1	Maria da Silva	F	2500.00
2	Pedro Pereira	M	990.00
3	Maria Cristina	F	1200.00
4	Antônio Carlos	M	990.00
5	Marcelo Augusto	M	1900.00
6	Pedro Silva	M	1050.00
7	Mônica da Silva	F	3000.00
8	Tiago Lima	M	1350.50

(8 row(s) affected)

Conteúdo de FUNCIONARIOS



## ■ Filtro de registros com **WHERE**

- O comando **SELECT** pode ser utilizado em conjunto com a cláusula **WHERE**.
- Isso permite que sejam aplicados filtros com base em determinados critérios. Sua sintaxe básica é dada por:

**SELECT** (Lista de Colunas)

**FROM** (Lista de Tabelas)

**WHERE** (Lista de Condições) 



## ■ Filtro de registros com **WHERE**

Operadores	Descrição
<code>=, &lt;, &gt;, &lt;=, &gt;=, &lt;&gt;, !=</code>	Operadores de comparação
<code>AND, OR, NOT</code>	Operadores lógicos
<code>BETWEEN</code>	Verifica se o valor do atributo está dentro de uma determinada faixa
<code>IS NULL</code>	Verifica se o valor do atributo é nulo
<code>LIKE</code>	Verifica se o valor do atributo coincide com determinado padrão de caracteres
<code>IN</code>	Verifica se o valor do atributo coincide com qualquer valor dentro de uma lista
<code>EXISTS</code>	Verifica se uma subconsulta retorna uma linha
<code>DISTINCT</code>	Limita os valores a valores exclusivos



## ■ Filtro de registros com **WHERE**

- Exibe as informações de todos os
- funcionários masculinos, cujo
- salário é maior do que 1000 reais

**SELECT** \* **FROM** FUNCIONARIOS

**WHERE** Sexo = 'M' AND  Lógico

Salario > 1000  Comparação

**GO**



## ■ Filtro de registros com **WHERE**

ID	Nome	Sexo	Admissao	Salario
---	-----	---	-----	-----
5	Marcelo Augusto	M	2017-12-09	1900.00
6	Pedro Silva	M	2015-11-15	1050.00
8	Tiago Lima	M	2016-05-10	1350.50

(3 row(s) affected)



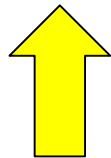
## ▪ Operador **NOT**

- Exibe os dados dos funcionários
- sexo seja diferente de 'M'
- Versão utilizando o operador !=

**SELECT** \* **FROM** FUNCIONARIOS

**WHERE** Sexo != 'M'

**GO**



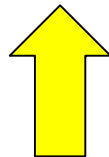
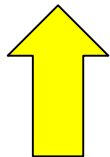
## ■ Operador **NOT**

- Exibe os dados dos funcionários
- sexo seja diferente de 'M'
- Versão utilizando o operador NOT

**SELECT** \* **FROM** FUNCIONARIOS

**WHERE** NOT Sexo = 'M'

**GO**





## ■ Operador **NOT**

ID	Nome	Sexo	Admissao	Salario
---	-----	---	-----	-----
1	Maria da Silva	F	2018-01-10	2500.00
3	Maria Cristina	F	2015-09-10	1200.00
7	Mônica da Silva	F	2014-10-12	3000.00

(3 row(s) affected)



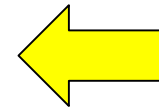
## ■ Operador **BETWEEN**

- Exibe os dados dos funcionários
- cujo salário esteja entre 1000
- e 2000 reais (inclusive). Utiliza
- os operadores  $\geq$ , AND e  $\leq$

**SELECT** \* **FROM** FUNCIONARIOS

**WHERE** Salario  $\geq$  1000 **AND**

Salario  $\leq$  2000



**GO**



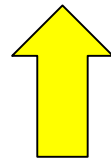
## ■ Operador **BETWEEN**

- Exibe os dados dos funcionários
- cujo salário esteja entre 1000
- e 2000 reais (inclusive). Utiliza
- o operador BETWEEN

**SELECT** \* **FROM** FUNCIONARIOS

**WHERE** Salario **BETWEEN** 1000 **AND** 2000

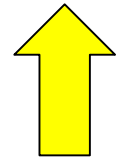
**GO**



## ■ Operador **BETWEEN**

ID	Nome	Sexo	Admissao	Salario
---	-----	---	-----	-----
3	Maria Cristina	F	2015-09-10	1200.00
5	Marcelo Augusto	M	2017-12-09	1900.00
6	Pedro Silva	M	2015-11-15	1050.00
8	Tiago Lima	M	2016-05-10	1350.50

(4 row(s) affected)





## ■ Operador EXISTS

- Verifica se existe algum funcionário cujo
- salário seja maior do que 10.000 reais e exibe
- uma mensagem personalizada.

```
IF EXISTS (SELECT * FROM FUNCIONARIOS  
           WHERE Salario > 10000)
```

```
    PRINT 'Alguém ganha mais que 10000  
    reais...'
```

```
ELSE
```

```
    PRINT 'Não localizamos ninguém...'
```

```
GO
```



- Consultas utilizando **LIKE**
  - O operador auxiliar **LIKE** é utilizado para verificar e comparar sequências de caracteres, dentro de um determinado campo.
  - Ele permite que sejam utilizados **caracteres curingas**, aumentando assim a capacidade de operação.



## ■ Caracteres coringas do operador **LIKE**

Caractere	Descrição
LIKE 'A%'	Valores que começam com a letra A.
LIKE '%ANA%'	Valores que tenham a letra ANA em qualquer posição.
LIKE '_A%'	Valores que tenham a letra A na segunda posição.
LIKE '1__'	Valores que começam com 1 e tenham 3 caracteres de comprimento.
LIKE '%6'	Valores que terminem com 6.
LIKE '_1%6'	Valores que tenham 1 na segunda posição e que termine com 6.
LIKE '_[O,I]%'	Valores que comecem com qualquer letra, tenham a letra O ou a letra I na segunda posição e terminem com qualquer letra.
LIKE '[^T]%'	Valores que comecem com qualquer letra, exceto T.



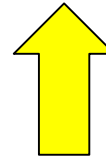
## ■ Operador **LIKE**

- Seleciona todos os funcionários
- cujo nome comece com a letra M.

**SELECT** \* **FROM** FUNCIONARIOS

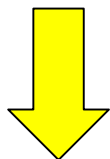
**WHERE** Nome **LIKE** 'M%'

**GO**





## ■ Operador **LIKE**



ID	Nome	Sexo	Admissao	Salario
---	-----	---	-----	-----
1	Maria da Silva	F	2018-01-10	2500.00
3	Maria Cristina	F	2015-09-10	1200.00
5	Marcelo Augusto	M	2017-12-09	1900.00
7	Mônica da Silva	F	2014-10-12	3000.00

(4 row(s) affected)



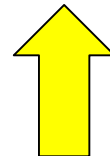
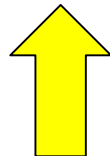
## ■ Operador **LIKE**

- Seleciona todos os funcionários
- cujo nome contenha 'SILVA'.
- Utiliza a função UPPER(), para
- converter o nome para maiúsculas.

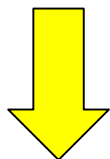
**SELECT** \* **FROM** FUNCIONARIOS

**WHERE** UPPER(Nome) **LIKE** '%SILVA%'

**GO**



## ■ Operador **LIKE**



ID	Nome	Sexo	Admissao	Salario
---	-----	---	-----	-----
1	Maria da Silva	F	2018-01-10	2500.00
6	Pedro Silva	M	2015-11-15	1050.00
7	Mônica da Silva	F	2014-10-12	3000.00

(3 row(s) affected)



- Consultas utilizando **IN**
  - O operador auxiliar **IN** possibilita realizar a busca de um valor específico dentro de uma lista de valores definidos, retornando **TRUE** caso o valor específico esteja na lista.



## ■ Operador **IN**

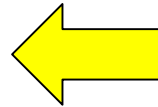
- Seleciona todos os funcionários
- cujo ID seja 1, 2 ou 5. Versão
- utilizando OR.

```
SELECT * FROM FUNCIONARIOS
```

```
WHERE ID = 1 OR
```

```
ID = 2 OR
```

```
ID = 5
```



GO



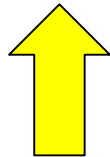
## ▪ Operador **IN**

- Seleciona todos os funcionários
- cujo ID seja 1, 2 ou 5. Versão
- utilizando IN.

```
SELECT * FROM FUNCIONARIOS
```

```
WHERE ID IN (1, 2, 5)
```

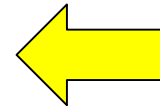
```
GO
```



## ■ Operador **IN**

- Utiliza uma subconsulta para
- retornar os valores da cláusula IN.

```
SELECT * FROM FUNCIONARIOS  
WHERE Salario IN  
    (SELECT Salario  
     FROM FUNCIONARIOS  
     WHERE Salario < 1000.00)
```



GO



## ■ Operador **IN**

ID	Nome	Sexo	Admissao	Salario
---	-----	---	-----	-----
2	Pedro Pereira	M	2015-03-25	990.00
4	Antônio Carlos	M	2015-05-15	990.00

(2 row(s) affected)





- Consultas utilizando **EXISTS**
  - O operador especial **EXISTS** pode ser utilizado sempre que for necessário executar um comando com base no resultado de outra consulta.
  - Ele retorna **TRUE** caso uma subconsulta retorne pelo menos alguma linha.



## ■ Operador EXISTS

- Verifica se algum funcionário
- possui o sobrenome 'Pereira'.

```
IF EXISTS (SELECT Nome  
           FROM FUNCIONARIOS  
           WHERE Nome LIKE '%Pereira%')  
PRINT 'Alguém chama Pereira...'  
GO
```



- Ordenando o resultado **ORDER BY**
  - A cláusula **ORDER BY** é utilizada para ordenar o resultado de um comando **SELECT**.
  - Ele é inserido após a cláusula **WHERE** e pode ser do tipo crescente (**ASC**) ou decrescente (**DESC**).



## ■ Cláusula **ORDER BY**

- Selecciona todos os funcionários,
- ordenando o resultado em ordem
- alfabética (A-Z).

**SELECT** \* **FROM** FUNCIONARIOS

**ORDER BY** Nome 

**GO**



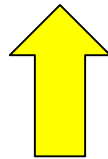
## ■ Cláusula **ORDER BY**

- Selecciona todos os funcionários,
- ordenando o resultado em ordem
- alfabética (A-Z).

**SELECT** \* **FROM** FUNCIONARIOS

**ORDER BY** Nome **ASC**

**GO**



## ■ Cláusula **ORDER BY**

- Selecciona todos os funcionários,
- ordenando o resultado em ordem
- alfabética (Z-A).

**SELECT** \* **FROM** FUNCIONARIOS

**ORDER BY** Nome **DESC**

**GO**



## ■ Cláusula **ORDER BY**



ID	Nome	Sexo	Admissao	Salario
---	-----	----	-----	-----
8	Tiago Lima	M	2016-05-10	1350.50
6	Pedro Silva	M	2015-11-15	1050.00
2	Pedro Pereira	M	2015-03-25	990.00
7	Mônica da Silva	F	2014-10-12	3000.00
1	Maria da Silva	F	2018-01-10	2500.00
3	Maria Cristina	F	2015-09-10	1200.00
5	Marcelo Augusto	M	2017-12-09	1900.00
4	Antônio Carlos	M	2015-05-15	990.00

(8 row(s) affected)



## ■ Cláusula **ORDER BY**

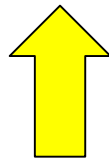
-- Insere dois novos funcionários

INSERT INTO FUNCIONARIOS VALUES

(9, 'Maria Cristina', 'F', '21/09/2012',  
1700.00),

(10, 'Maria Cristina', 'F', '10/10/2017',  
1400.00)

GO



Repare que os nomes são iguais!





## ■ Cláusula **ORDER BY**

- Seleciona todos os funcionários, ordenando o
- resultado primeiro pelo nome em ordem alfabética
- (A-Z) e depois pelos dados de quem têm o maior
- salário. Exibe os dados somente dos funcionários
- cujo salário seja menor do que 3000 reais.

```
SELECT * FROM FUNCIONARIOS
```

```
WHERE Salario < 3000
```

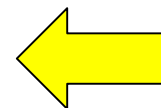
```
ORDER BY Nome,  
         Salario DESC
```

```
GO
```



## ■ Cláusula **ORDER BY**

ID	Nome	Sexo	Admissao	Salario
---	-----	----	-----	-----
4	Antônio Carlos	M	2015-05-15	990.00
5	Marcelo Augusto	M	2017-12-09	1900.00
9	Maria Cristina	F	2012-09-21	1700.00
10	Maria Cristina	F	2017-10-10	1400.00
3	Maria Cristina	F	2015-09-10	1200.00
1	Maria da Silva	F	2018-01-10	2500.00
2	Pedro Pereira	M	2015-03-25	990.00
6	Pedro Silva	M	2015-11-15	1050.00
8	Tiago Lima	M	2016-05-10	1350.50



(9 row(s) affected)



- **Na próxima aula veremos**
  - Alteração da estrutura de tabelas.

