



# **DISCIPLINA: Banco de Dados 1**

Prof. **GIOVANI** Volnei Meinerz

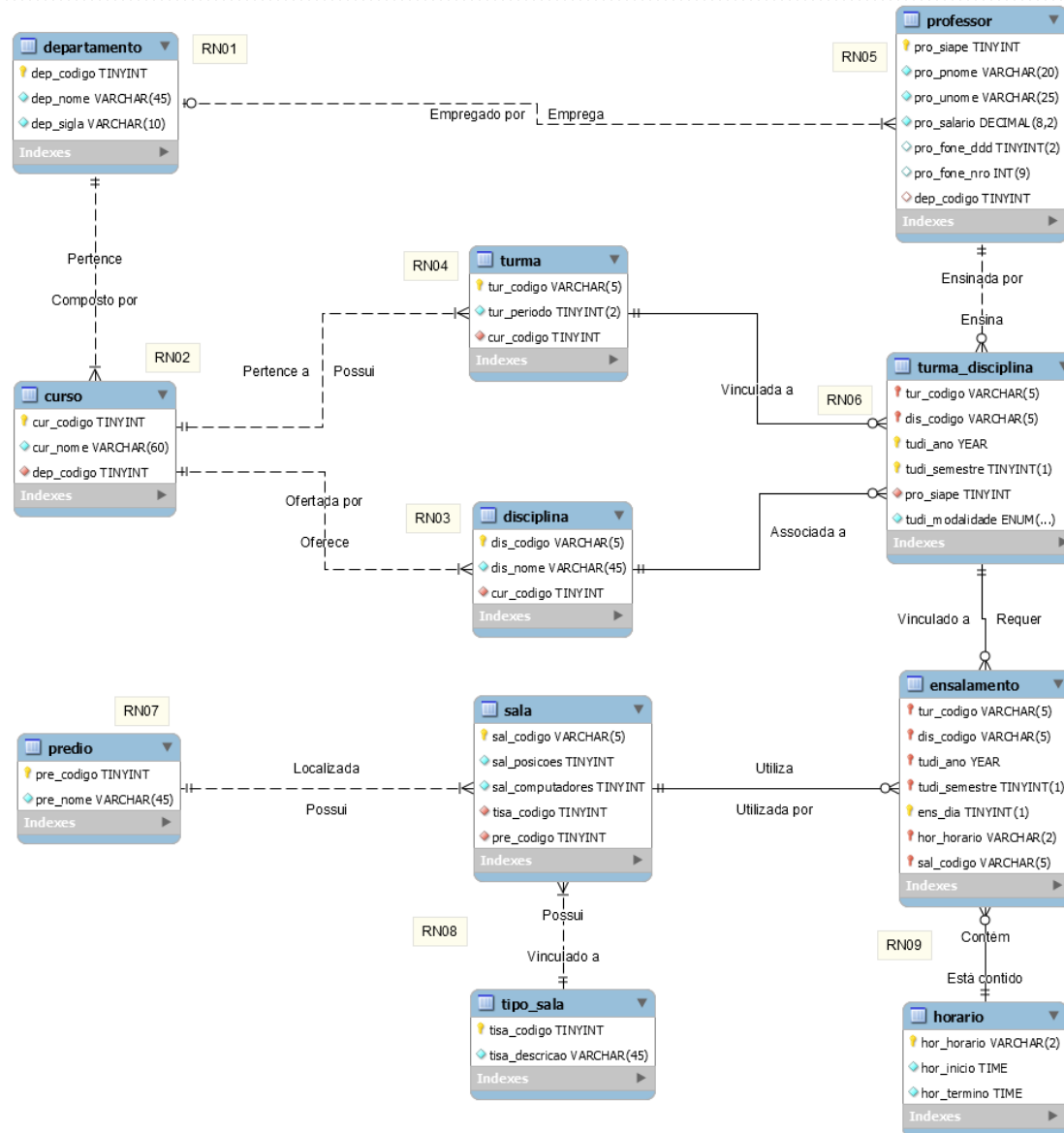
Aula 20 – SQL (cont.)

# Objetivos da Aula

---

- Aprender os diferentes tipos de subconsultas e sua aplicação

# Cenário



# Subconsultas Aninhadas

---

- Frequentemente é necessário processar dados com base em outros dados processados previamente, cujo resultado normalmente é desconhecido
- Nesses casos, a SQL permite utilizar subconsultas para gerar um resultado necessário e, então, utilizá-lo na consulta mais externa

# Subconsultas Aninhadas (cont.)

---

- Características das subconsultas
  - É uma consulta no interior de outra consulta
  - A primeira consulta é conhecida como consulta externa
  - A consulta no interior é conhecida como consulta interna
  - A consulta interna é executada primeiro
  - A saída de uma consulta interna é utilizada como entrada da consulta externa

# Subconsultas Aninhadas (cont.)

---

- Subconsultas podem ser utilizadas com cláusulas **SELECT**, **CREATE TABLE**, **INSERT**, **UPDATE** e **DELETE** em conjunto com os seguintes operadores
  - =, <, >, >=, <=, <>, !=
  - **IN, NOT IN**
    - Teste de Membros de Conjunto
  - **ALL, ANY (SOME)**
    - Comparação de Conjuntos
  - **EXISTS, NOT EXISTS**
    - Teste de Relações Vazias

# Subconsultas Aninhadas (cont.)

→ Como implementar e usar subconsultas

→ Problema

→ Qual é o nome do professor com o menor salário?

```
11  -- Nomes e salários dos professores
12  •  SELECT
13      pro_pnome, pro_unome, pro_salario
14  FROM
15      professor
16  ORDER BY pro_unome;
```

pro_pnome	pro_unome	pro_salario
Corinna	Enoellmann	130000.00
Tilo	Gerhold	65000.00
Gerhard	Huettio	85000.00
Anoela	Lehmann	75000.00
Mano	Lima	81000.00
Cenair	Maicá	89000.00
Luiz	Marengo	67000.00
Giovani	Meinerz	84000.00
Lisa	Reimann	95000.00
Lena	Reimann	145000.00
Hanka	Ruebbert	60000.00
Ekkehart	Schubbert	55000.00
Manfred	Schubbert	79000.00

# Operador de Comparação

- Solução: Resolvendo o problema em 2 passos
  - Qual é o valor do menor salário?

```
21 -- Valor do menor salário
22 • SELECT
23     MIN(pro_salario) 'Menor Salário'
24 FROM
25     professor;
```

	Menor Salário
	55000.00

- Qual é o nome da(s) pessoa(s) a que esse salário corresponde?

```
29 -- Nome do professor com o menor salário
30 • SELECT
31     pro_pnome Nome, pro_unome Sobrenome
32 FROM
33     professor
34 WHERE
35     pro_salario = 55000;
```

	Nome	Sobrenome
	Ekkehart	Schubbert



# Operador de Comparação (cont.)

- Solução: Resolvendo por meio de subconsulta

**SELECT externo**

```
39 -- Nome do professor com o menor salário
40 • SELECT pro_pnome, pro_unome
41   FROM professor
42  WHERE pro_salario = (SELECT MIN(pro_salario)
43                      FROM professor);
```

**SELECT interno**

**Operador de comparação**

	pro_pnome	pro_unome
	Ekkehart	Schubbert

## → Nota

- O cabeçalho é composto pelas colunas do **SELECT** externo
- O resultado do **SELECT** externo depende do resultado do **SELECT** interno

# Operador de Comparação (cont.)

## ➤ Exemplo

- Encontre e mostre os nomes e os salários de todos os professores, cujo salário esteja acima da média salarial dos professores. Ordene pelo salário, do menor para o maior.

```
58 -- Média salarial dos professores
59 • SELECT AVG(pro_salario) AS 'Média Salarial Professores'
60 FROM professor;
```

Média Salarial Professores
----------------------------

85384.615385
--------------

```
62 -- Professores cujo salário seja superior a média salarial
63 • SELECT pro_pnome, pro_unome, pro_salario
64 FROM professor
65 WHERE pro_salario > (SELECT AVG(pro_salario) FROM professor)
66 ORDER BY pro_salario;
```

pro_pnome	pro_unome	pro_salario
Cenair	Maicá	89000.00
Lisa	Reimann	95000.00
Corinna	Enoellmann	130000.00
Lena	Reimann	145000.00

# Subconsultas no SELECT e WHERE

- Exemplo – **subconsulta como expressão de seleção**
  - Encontre e mostre os nomes e os salários de todos os professores, cujo salário esteja acima da média salarial dos professores, mostrando, também, a média salarial

Operador de comparação

Subconsulta no **SELECT**

```
68 -- Professores cujo salário seja superior a média salarial, mostrando, também, a média salarial
69 • SELECT pro_pnome, pro_unome, pro_salario, (SELECT AVG(pro_salario) FROM professor) AS 'Média Salarial'
70 FROM professor
71 WHERE pro_salario > (SELECT AVG(pro_salario) FROM professor)
72 ORDER BY pro_salario;
```

Subconsulta no **WHERE**

pro_pnome	pro_unome	pro_salario	Média Salarial
Cenair	Maicá	89000.00	85384.615385
Lisa	Reimann	95000.00	85384.615385
Corinna	Enoellmann	130000.00	85384.615385
Lena	Reimann	145000.00	85384.615385

# Subconsultas no FROM

- ➔ Exemplo – **subconsulta como tabela na cláusula FROM**
- ➔ Encontre e mostre os professores (seus nomes) sem telefone cadastrado

```
10 -- Professores sem telefone cadastrado
11 -- Usando subconsulta como tabela na cláusula FROM
12 • SELECT pro_pnome Nome, pro_unome Sobrenome
13 FROM (SELECT *
14       FROM professor
15       WHERE pro_fone_nro IS NULL) AS professor_fones;
```

Nome	Sobrenome
Tilo	Gerhold
Lisa	Reimann

```
1 • SELECT
2     pro_pnome Nome,
3     pro_unome Sobrenome,
4     pro_fone_ddd DDD,
5     pro_fone_nro Fone
6 FROM
7     professor;
```

Nome	Sobrenome	DDD	Fone
Hanka	Ruebbert	43	35239000
Tilo	Gerhold	NULL	NULL
Ekkehart	Schubbert	11	754210000
Gerhard	Huettig	14	33448888
Angela	Lehmann	43	35237777
Lisa	Reimann	NULL	NULL
Corinna	Engellmann	55	995554500
Manfred	Schubbert	43	998456587
Lena	Reimann	14	997465544
Giovani	Meinerz	55	999838457
Luiz	Marenco	55	996814596
Mano	Lima	55	975824684
Cenair	Maicá	55	997896341

# Subconsultas com Operador IN

## Teste de Membros de Conjunto

- Operador [NOT] IN
  - Utilizado para verificar se o valor de um atributo coincide com qualquer valor de uma lista (está contido, ou não)
- Sintaxe SQL

```
SELECT A1 [, A2, ... , An]  
FROM T1 [, T2, ... , Tn]  
[WHERE Expressão [NOT] IN (Valor1, Valor2, ... , Valorn)]
```

- Onde
  - **Expressão**, valor de atributo a ser testado
  - **Valor<sub>1</sub>, Valor<sub>2</sub>, ..., Valor<sub>n</sub>**, conjunto de valores em relação aos quais a **Expressão** será testada

# Subconsultas com Operador IN (cont.)

---

- Vimos que, para **Teste de Membros de Conjunto**, pode ser utilizado o operador especial **[NOT] IN**, que permite verificar se o valor de um atributo é igual a um dos valores contidos em um determinado conjunto
- O operador **[NOT] IN** é especialmente útil quando utilizado com subconsultas

# Subconsultas com Operador IN (cont.)

- Ao empregar o operador **IN**, a consulta é escrita na forma de blocos **SELECT-FROM-WHERE** aninhados
- Problema
  - Encontrar quais turmas/disciplinas foram ensaladas em quais salas de um determinado prédio/bloco. Mostre turma, sala, ano e semestre

```
40 • SELECT
41     tur_codigo, e.sal_codigo, tudi_ano, tudi_semestre
42 FROM
43     ensalamento e
44 WHERE
45     e.sal_codigo IN (SELECT sal_codigo
46                     FROM sala s, predio p
47                     WHERE s.pre_codigo = p.pre_codigo
48                          AND pre_nome = 'Bloco A');
```

tur_codigo	sal_codigo	tudi_ano	tudi_semestre
ES21	A040	2017	1
C41	A040	2017	2
C41	A040	2017	2

# Subconsultas com Operador IN (cont.)

## ✈ Entendendo cada bloco...

```
1 • SELECT
2     tur_codigo, dis_codigo, tudi_ano, tudi_semestre
3 FROM
4     ensalamento;
```

SELECT externo

tur_codigo	sal_codigo	tudi_ano	tudi_semestre
M31	P105	2017	2
M31	P105	2017	2
ES21	I205	2017	2
N12A	I205	2017	2
N12A	I205	2017	2
N12A	I205	2017	2
N12A	I205	2017	2
E21	K009	2017	2
E21	K009	2017	2
ES21	A040	2017	1
C41	A040	2017	2
C41	A040	2017	2

```
38 -- Encontrar quais turmas/disciplinas foram ensaladas em quais salas de um determinado prédio/bloco
39 -- Mostre a turma, a sala, ano e semestre
40 • SELECT
41     tur_codigo, e.sal_codigo, tudi_ano, tudi_semestre
42 FROM
43     ensalamento e
44 WHERE
45     e.sal_codigo IN (SELECT sal_codigo
46                     FROM sala s, predio p
47                     WHERE s.pre_codigo = p.pre_codigo
48                          AND pre_nome = 'Bloco A');
```

SELECT interno

```
54 • SELECT sal_codigo, pre_nome
55 FROM sala s, predio p
56 WHERE s.pre_codigo = p.pre_codigo AND pre_nome = 'Bloco A'
```

sal_codigo	pre_nome
A040	Bloco A
A137	Bloco A
A140	Bloco A
A146	Bloco A



# Subconsultas com Operador IN (cont.)

## ➤ Problema

- Encontrar os professores (mostrar seus nomes) aos quais nenhuma turma tenha sido atribuída

```
3 SELECT pro_pnome, pro_unome
4 FROM professor p
5 WHERE pro_pnome NOT IN (SELECT pro_pnome
6                           FROM turma_disciplina td
7                           WHERE td.pro_siape = p.pro_siape);
```

pro_pnome	pro_unome
Tilo	Gerhold
Ekkehart	Schubbert
Manfred	Schubbert
Lena	Reimann
Giovani	Meinerz

# Subconsultas com Operador ALL e ANY

---

- Operadores **ALL** e **ANY** (**SOME** tem o mesmo efeito)
- Permitem realizar a **Comparação de Conjuntos de Valores**
- Combinados com operadores de comparação
  - **>, >=, <, <=, <>, !=**

# Subconsultas com Operador ALL e ANY (cont.)

## ➤ Exemplo – com ANY

- Encontre os professores cujo salário é maior que o salário de pelo menos um dos professores do DACOM. Mostre nome e salário. Ordene de forma ascendente pelo salário

```
5 • SELECT pro_pnome, pro_unome, pro_salario
6   FROM professor
7  WHERE pro_salario > ANY (SELECT pro_salario
8                           FROM departamento d, professor p
9                           WHERE p.dep_codigo = d.dep_codigo and dep_sigla = 'DACOM')
10  ORDER BY pro_salario;
```

“maior que pelo menos um”

pro_pnome	pro_unome	pro_salario
Gerhard	Huettig	85000.00
Cenair	Maicá	89000.00
Lisa	Reimann	95000.00
Corinna	Endellmann	130000.00
Lena	Reimann	145000.00

# Subconsultas com Operador ALL e ANY (cont.)

- Exemplo – com ANY (cont.)
- Entendendo o que cada bloco gera...

pro_pnome	pro_unome	pro_salario
Ekkehart	Schubbert	55000.00
Hanka	Ruebbert	60000.00
Tilo	Gerhold	65000.00
Luiz	Marengo	67000.00
Anoela	Lehmann	75000.00
Manfred	Schubbert	79000.00
Mano	Lima	81000.00
Giovani	Meinerz	84000.00
Gerhard	Huettig	85000.00
Cenair	Maicá	89000.00
Lisa	Reimann	95000.00
Corinna	Endellmann	130000.00
Lena	Reimann	145000.00

pro_pnome	pro_unome	pro_salario
Giovani	Meinerz	84000.00
Lisa	Reimann	95000.00
Corinna	Endellmann	130000.00

```
5 • SELECT pro_pnome, pro_unome, pro_salario
6 FROM professor
7 WHERE pro_salario > ANY (SELECT pro_salario
8                           FROM departamento d, professor p
9                           WHERE p.dep_codigo = d.dep_codigo and dep_sigla = 'DACOM')
10 ORDER BY pro_salario;
```

pro_pnome	pro_unome	pro_salario
Gerhard	Huettig	85000.00
Cenair	Maicá	89000.00
Lisa	Reimann	95000.00
Corinna	Endellmann	130000.00
Lena	Reimann	145000.00

# Subconsultas com Operador ALL e ANY (cont.)

## → Exemplo – com ALL

- Encontre os professores cujo salário é maior que o salário de todos os professores do DACOM. Mostre nome e salário. Ordene de forma ascendente pelo salário

“maior que todos”

```
33 • SELECT pro_pnome, pro_unome, pro_salario
34 FROM professor
35 WHERE pro_salario > ALL (SELECT pro_salario
36                           FROM departamento d, professor p
37                           WHERE p.dep_codigo = d.dep_codigo and dep_sigla = 'DACOM')
38 ORDER BY pro_salario;
```

pro_pnome	pro_unome	pro_salario
Lena	Reimann	145000.00

# Subconsultas com Operador EXISTS

- ➔ Para **Teste de Relações Vazias**
- ➔ Operador **[NOT] EXISTS**
  - ➔ Permite testar se existe, ou não, alguma tupla no resultado de uma subconsulta
- ➔ Problema
  - ➔ Encontre e mostre os departamentos (sigla) aos quais não há professor vinculado

```
2 • SELECT dep_sigla
3   FROM departamento d
4  WHERE NOT EXISTS (SELECT * FROM professor p
5                    WHERE p.dep_codigo = d.dep_codigo);
```

dep_sigla
DAELN
DAELT
DAFIS
DAGEE

# Subconsultas com Operador EXISTS (cont.)

- Problema (entendendo o que cada bloco gera)
  - Encontre e mostre os departamentos (sigla) aos quais não há professor vinculado

dep_sigla
DACOM
DAELE
DAMAT
DAELN
DAELT
DAMEC
DAFIS
DAGEE

dep_sigla
DACOM
DAELE
DAMAT
DAMEC

```
2 • SELECT dep_sigla
3   FROM departamento d
4   WHERE NOT EXISTS (SELECT * FROM professor p
5                      WHERE p.dep_codigo = d.dep_codigo);
```

dep_sigla
DAELN
DAELT
DAFIS
DAGEE

# Subconsultas com Operador EXISTS (cont.)

## ➤ Exemplo

- Encontre as salas nas quais não foram ensaladas turmas/disciplinas. Mostre o código da sala.

sal_codigo
A040
A137
A140
A146
I201
I202
I205
K005
K008
K009
P003
P005
P101
P105
P205

```
18 • SELECT sal_codigo
19 FROM sala s
20 WHERE NOT EXISTS (SELECT *
21                     FROM ensalamento e
22                     WHERE e.sal_codigo = s.sal_codigo);
```

sal_codigo
P105
I205
K009
A040

sal_codigo
I201
I202
K005
K008
P003
P005
P101
P205
A137
A140
A146



# CREATE TABLE ... SELECT

- SQL permite criar uma **nova tabela** com base em linhas e colunas selecionadas de uma ou mais **tabelas existentes**

## → Sintaxe SQL

```
CREATE TABLE nova_tabela  
  AS (SELECT A1 [, A2, ... , An]  
      FROM tabela_existente(s)  
      [WHERE P]  
  );
```

# CREATE TABLE ... SELECT

→ Exemplo – copiar todo o conteúdo de uma tabela

→ Estrutura e conteúdo da tabela a ser copiada

Table: turma\_disciplina

Columns:

<u>tur_codigo</u>	varchar(5) PK
<u>dis_codigo</u>	varchar(5) PK
<u>tudi_ano</u>	year(4) PK
<u>tudi_semestre</u>	tinyint(1) PK
<u>pro_siape</u>	tinyint(4)
tudi_modalidade	enum('Presencial','Distância','Semipresencial')

tur_codigo	dis_codigo	tudi_ano	tudi_semestre	pro_siape	tudi_modalidade
C41	EC34D	2017	2	6	Presencial
C51	EC35B	2017	1	12	Presencial
C51	EC35B	2017	2	13	Presencial
E21	MA35B	2017	2	4	Presencial
ES21	IF62H	2017	1	11	Presencial
ES21	IF62H	2017	2	11	Presencial
ES31	IF63C	2016	1	5	Presencial
M31	EM33H	2017	2	1	Presencial
N12A	AN32C	2017	2	11	Presencial
N12B	AN32C	2017	2	7	Presencial
N12B	AN32C	2018	1	7	Distância
N12SP	AN32C	2017	2	11	Semipresencial
NULL	NULL	NULL	NULL	NULL	NULL

# CREATE TABLE ... SELECT

- ➔ Exemplo – copiar **todo** o conteúdo de uma tabela (cont.)
  - ➔ Criar uma nova tabela, a partir da tabela **turma\_disciplina**, copiando **todas as linhas** de **todas as colunas** da tabela

```
4 • CREATE TABLE tda
5 AS (SELECT * FROM turma_disciplina);
```

Table: **tda**

Columns:

tur_codigo	varchar(5)
dis_codigo	varchar(5)
tudi_ano	year(4)
tudi_semestre	tinyint(1)
pro_siape	tinyint(4)
tudi_modalidade	enum('Presencial','Distância','Semipresencial')

Table: **turma\_disciplina**

Columns:

<u>tur_codigo</u>	varchar(5) PK
<u>dis_codigo</u>	varchar(5) PK
<u>tudi_ano</u>	year(4) PK
<u>tudi_semestre</u>	tinyint(1) PK
pro_siape	tinyint(4)
tudi_modalidade	enum('Presencial','Distância','Semipresencial')

- ➔ **Nota:** nenhuma restrição de integridade de chave e referencial é aplicada automaticamente à nova tabela

# CREATE TABLE ... SELECT

- Exemplo – copiar **parte** do conteúdo de uma tabela
- Criar uma nova tabela, a partir da tabela **turma\_disciplina**, copiando **todas as linhas** de dados de **algumas colunas**

```
10 • CREATE TABLE tdb
11   AS (SELECT tur_codigo, tudi_ano, tudi_semestre, pro_siape
12        FROM turma_disciplina);
```

Table: **tdb**

Columns:

tur_codigo	varchar(5)
tudi_ano	year(4)
tudi_semestre	tinyint(1)
pro_siape	tinyint(4)

Table: **turma\_disciplina**

Columns:

<u>tur_codigo</u>	varchar(5) PK
<u>dis_codigo</u>	varchar(5) PK
<u>tudi_ano</u>	year(4) PK
<u>tudi_semestre</u>	tinyint(1) PK
<u>pro_siape</u>	tinyint(4)
tudi_modalidade	enum('Presencial', 'Distância', 'Semipresencial')

# CREATE TABLE ... SELECT

- Exemplo – copiar **parte** do conteúdo de **múltiplas** tabelas
- Criar uma nova tabela e copiar **parte das linhas** de dados de **algumas colunas** de múltiplas tabelas, modificando o nome das colunas na nova tabela

Table: **professor**

Columns:

<u>pro_siape</u>	tinyint(4) PK
pro_pnome	varchar(20)
pro_unome	varchar(25)
pro_salario	decimal(8,2)
pro_fone_ddd	tinyint(2)
pro_fone_nro	int(9)
dep_codigo	tinyint(4)

Table: **disciplina**

Columns:

<u>dis_codigo</u>	varchar(5) PK
dis_nome	varchar(45)
cur_codigo	tinyint(4)

Table: **turma\_disciplina**

Columns:

<u>tur_codigo</u>	varchar(5) PK
<u>dis_codigo</u>	varchar(5) PK
<u>tudi_ano</u>	year(4) PK
<u>tudi_semestre</u>	tinyint(1) PK
pro_siape	tinyint(4)
tudi_modalidade	enum('Presencial','Distância','Semipresencial')

# CREATE TABLE ... SELECT

- Exemplo – copiar **parte** do conteúdo de **múltiplas** tabelas
  - Criar uma nova tabela e copiar **parte das linhas** de dados de **algumas colunas** de múltiplas tabelas, modificando o nome das colunas na nova tabela (cont.)

```
20 • CREATE TABLE turma2017_01sem
21 AS (SELECT td.tur_codigo AS Turma,
22         d.dis_nome AS Disciplina,
23         p.pro_pnome AS Professor,
24         p.pro_salario AS Salario,
25         td.tudi_ano AS Ano,
26         td.tudi_semestre AS Semestre,
27         td.tudi_modalidade AS Modalidade
28 FROM turma_disciplina td, disciplina d, professor p
29 WHERE td.dis_codigo = d.dis_codigo AND
30        td.pro_siape = p.pro_siape AND
31        td.tudi_ano = 2017 AND
32        td.tudi_semestre = 1 AND
33        td.tudi_modalidade = 'Presencial');
```

Table: turma2017 01sem

**Columns:**

Turma	varchar(5)
Disciplina	varchar(45)
Professor	varchar(20)
Salario	decimal(8,2)
Ano	year(4)
Semestre	tinyint(1)
Modalidade	enum('Presencial','Distância','Semipresencial')

```
1 ● SELECT * FROM reservas.turma2017 01sem;
```

Turma	Disciplina	Professor	Salario	Ano	Semestre	Modalidade
C51	Banco de Dados 2	Mano	81000.00	2017	1	Presencial
ES21	Banco De Dados 1	Luiz	67000.00	2017	1	Presencial

# INSERT ... SELECT

- SQL permite adicionar linhas a uma tabela, utilizando outra tabela como fonte de dados
- Ou seja, é possível inserir linhas com base no resultado de uma consulta
- Sintaxe SQL

```
INSERT INTO nome_da_tabela (A1, A2, ... ,An)  
  (SELECT A1 [, A2, ... ,An]  
    FROM tabela_existente(s)  
    [WHERE P]);
```

# INSERT ... SELECT

## → Exemplo

- Selecionar dados de 3 tabelas e inseri-los em uma nova tabela
- Tabelas das quais os dados serão selecionados

Table: **turma\_disciplina**

Columns:

<u>tur_codigo</u>	varchar(5) PK
<u>dis_codigo</u>	varchar(5) PK
<u>tudi ano</u>	year(4) PK
<u>tudi semestre</u>	tinyint(1) PK
<u>pro_siape</u>	tinyint(4)
tudi_modalidade	enum('Presencial','Distância','Semipresencial')

Table: **disciplina**

Columns:

<u>dis_codigo</u>	varchar(5) PK
dis_nome	varchar(45)
<u>cur_codigo</u>	tinyint(4)

Table: **professor**

Columns:

<u>pro_siape</u>	tinyint(4) PK
pro_pnome	varchar(20)
pro_unome	varchar(25)
pro_salario	decimal(8,2)
pro_fone_ddd	tinyint(2)
pro_fone_nro	int(9)
<u>dep_codigo</u>	tinyint(4)



# INSERT ... SELECT

## → Exemplo (cont.)

### → Criando a nova tabela...

```
1  -- Tabela criada para nela inserir as turmas/disciplinas, ministradas
2  -- pelos professores aos quais foram atribuídas, da modalidade
3  -- "Presencial", do segundo semestre de 2017
4  CREATE TABLE turma2017_02sem (
5      Turma varchar(5) NOT NULL,
6      Disciplina varchar(45) NOT NULL,
7      Professor varchar(20) NOT NULL,
8      Ano year(4) NOT NULL,
9      Semestre tinyint(1) NOT NULL,
10     Modalidade enum('Presencial','Distância','Semipresencial') NOT NULL
11 );
```

Table: **turma2017\_02sem**

**Columns:**

Turma	varchar(5)
Disciplina	varchar(45)
Professor	varchar(20)
Ano	year(4)
Semestre	tinyint(1)
Modalidade	enum('Presencial','Distância','Semipresencial')

→ Os nomes das colunas, bem como a quantidade de colunas não precisam ser iguais aos das tabelas das quais os dados serão selecionados

# INSERT ... SELECT

## → Exemplo (cont.)

### → Inserindo os dados na tabela...

```
25 • INSERT INTO turma2017_02sem
26   (SELECT td.tur_codigo,
27         d.dis_nome,
28         p.pro_pnome,
29         td.tudi_ano,
30         td.tudi_semestre,
31         td.tudi_modalidade
32   FROM turma_disciplina td, disciplina d, professor p
33   WHERE td.dis_codigo = d.dis_codigo AND
34         td.pro_siape = p.pro_siape AND
35         td.tudi_ano = 2017 AND
36         td.tudi_semestre = 2 AND
37         td.tudi_modalidade = 'Presencial');
```

Table: turma2017\_02sem

**Columns:**

Turma	varchar(5)
Disciplina	varchar(45)
Professor	varchar(20)
Ano	year(4)
Semestre	tinyint(1)
Modalidade	enum('Presencial','Distância','Semipresencial')

- A listagem das colunas no **INSERT** é opcional, desde que os valores devolvidos pelo **SELECT** estejam na mesma ordem em que os atributos estão listados na tabela
- Características dos dados (domínio) devolvidos pelo **SELECT** devem coincidir com o domínio das colunas da tabela

# INSERT ... SELECT


## → Exemplo (cont.)


### → Resultado do **INSERT** a partir do **SELECT**

39 ● **SELECT \* FROM turma2017\_02sem;**


<

Result Grid





Filter Rows:

Export: 

Wrap Cell Cont

	Turma	Disciplina	Professor	Ano	Semestre	Modalidade
	C41	Banco De Dados 1	Lisa	2017	2	Presencial
	C51	Banco de Dados 2	Cenair	2017	2	Presencial
	E21	Probabilidade Estatística	Gerhard	2017	2	Presencial
	ES21	Banco De Dados 1	Luiz	2017	2	Presencial
	M31	Física Experimental	Hanka	2017	2	Presencial
	N12A	Banco De Dados 1	Luiz	2017	2	Presencial
	N12B	Banco De Dados 1	Corinna	2017	2	Presencial

# UPDATE ... SELECT

## Modificar com SQL

- O comando **UPDATE** permite modificar valores já existentes em atributos de uma tabela
- Sintaxe SQL

```
UPDATE nome_da_tabela  
SET Atributo1 = Expressão1 [, Consulta1]  
    Atributon = Expressãon [, Consultan]  
[WHERE Condição];
```

- É possível modificar um ou mais atributos
- A cláusula **WHERE** é opcional
  - Se especificada, modificará a tupla que atender a condição
  - Se omitida, todas as tuplas serão modificadas

# UPDATE ... SELECT

## ➤ Exemplo

- Na tabela **turma2017\_01sem**, modifique o salário de todos os professores, de forma que a eles seja atribuído o maior salário recebido dentre os professores (tabela **professor**)

Table: **turma2017\_01sem**

**Columns:**

Turma	varchar(5)
Disciplina	varchar(45)
Professor	varchar(20)
Salario	decimal(8,2)
Ano	year(4)
Semestre	tinyint(1)
Modalidade	enum('Presencial','Distância','Semipresencial')

```
1 • SELECT * FROM reservas.turma2017_01sem;
```

Turma	Disciplina	Professor	Salario	Ano	Semestre	Modalidade
C51	Banco de Dados 2	Mano	81000.00	2017	1	Presencial
ES21	Banco De Dados 1	Luiz	67000.00	2017	1	Presencial

```
14 -- Atribui maior salário dentre os professores
15 • UPDATE turma2017_01sem
16 SET Salario = (SELECT MAX(pro_salario) FROM professor);
```

```
1 • SELECT * FROM reservas.turma2017_01sem;
```

Turma	Disciplina	Professor	Salario	Ano	Semestre	Modalidade
C51	Banco de Dados 2	Mano	145000.00	2017	1	Presencial
ES21	Banco De Dados 1	Luiz	145000.00	2017	1	Presencial

# UPDATE ... SELECT

## ➔ Exemplo

- ➔ Na tabela **turma2017\_01sem**, modifique o salário dos professores, cujo nome termine com “z”, de forma que a estes seja atribuído o menor salário recebido dentre os professores

Table: **turma2017\_01sem**

**Columns:**

Turma	varchar(5)
Disciplina	varchar(45)
Professor	varchar(20)
Salario	decimal(8,2)
Ano	year(4)
Semestre	tinyint(1)
Modalidade	enum('Presencial','Distância','Semipresencial')

```
1 • SELECT * FROM reservas.turma2017_01sem;
```

Turma	Disciplina	Professor	Salario	Ano	Semestre	Modalidade
C51	Banco de Dados 2	Mano	145000.00	2017	1	Presencial
ES21	Banco De Dados 1	Luiz	145000.00	2017	1	Presencial

```
18 -- Atribui menor salário aos professores cujo nome termine com "z"
19 • UPDATE turma2017_01sem
20 SET Salario = (SELECT MIN(pro_salario) FROM professor)
21 WHERE Professor LIKE '%z';
```

```
1 • SELECT * FROM reservas.turma2017_01sem;
```

Turma	Disciplina	Professor	Salario	Ano	Semestre	Modalidade
C51	Banco de Dados 2	Mano	145000.00	2017	1	Presencial
ES21	Banco De Dados 1	Luiz	55000.00	2017	1	Presencial

# DELETE ... SELECT

## Excluir com SQL

- O comando **DELETE** permite excluir registros de uma tabela
  - Exclui registros inteiros
  - Não permite excluir apenas valores de atributos particulares

### → Sintaxe SQL

```
DELETE FROM nome_da_tabela  
[WHERE Condição];
```

- A cláusula **WHERE** pode ser omitida quando todos os registros da tabela precisarem ser excluídos

# DELETE ... SELECT

---

## → Sintaxe SQL

```
DELETE FROM nome_da_tabela  
[WHERE Condição];
```

- O predicado de um **WHERE** em um comando **DELETE** pode ser tão complexo quanto em um comando **SELECT**
- Embora possamos excluir tuplas apenas de uma tabela de cada vez, podemos referenciar qualquer quantidade de tabelas em um **SELECT-FROM-WHERE** aninhado na cláusula **WHERE** de um **DELETE**



# DELETE ... SELECT

## ➤ Exemplo

- Exclua todos os professores aos quais nenhuma turma/disciplina tenha sido atribuída

```
1  -- Professores cadastrados
2  • SELECT pro_siape, pro_pnome, pro_unome
3  FROM professor;
```

pro_siape	pro_pnome	pro_unome
1	Hanka	Ruebbert
2	Tilo	Gerhold
3	Ekkehart	Schubbert
4	Gerhard	Huettig
5	Anoela	Lehmann
6	Lisa	Reimann
7	Corinna	Endellmann
8	Manfred	Schubbert
9	Lena	Reimann
10	Giovani	Meinerz
11	Luiz	Marengo
12	Mano	Lima
13	Cenair	Maicá

```
5  -- Professores aos quais nenhuma turma/disciplina foi atribuída
6  • SELECT pro_siape, pro_pnome, pro_unome
7  FROM professor p
8  WHERE pro_siape NOT IN (SELECT pro_siape
9                           FROM turma_disciplina td
10                          WHERE td.pro_siape = p.pro_siape);
```

pro_siape	pro_pnome	pro_unome
2	Tilo	Gerhold
3	Ekkehart	Schubbert
8	Manfred	Schubbert
9	Lena	Reimann
10	Giovani	Meinerz

# DELETE ... SELECT

## ➤ Exemplo (cont.)

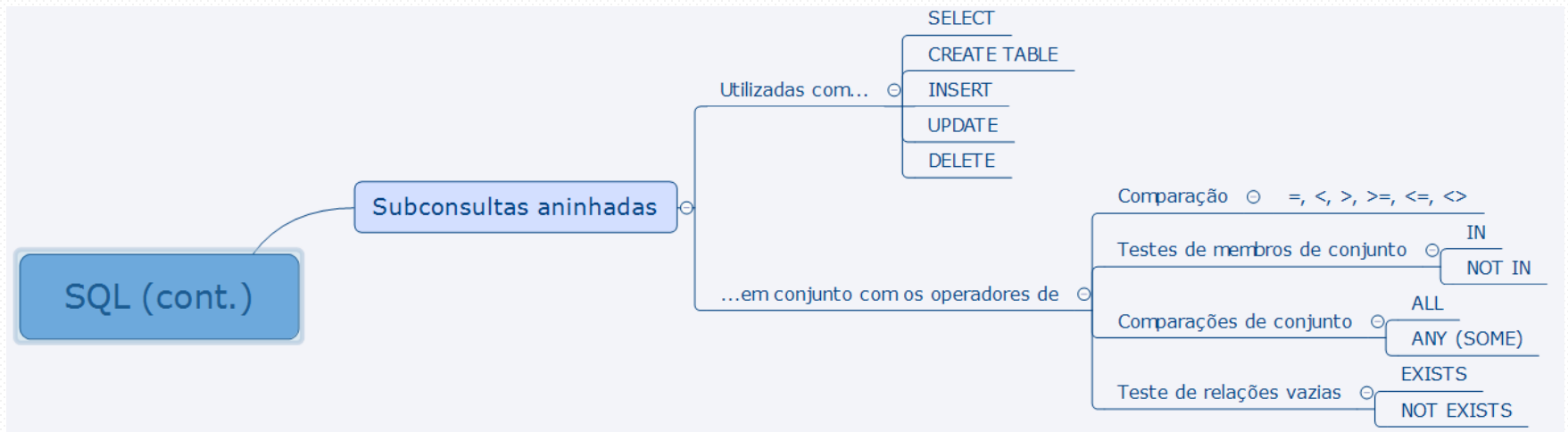
- Exclua todos os professores aos quais nenhuma turma tenha sido atribuída

```
12 -- Exclui os professores aos quais nenhuma turma/disciplina tenha sido atribuída
13 • DELETE FROM professor
14   WHERE NOT EXISTS (SELECT *
15                     FROM turma_disciplina
16                     WHERE turma_disciplina.pro_siape = professor.pro_siape);
```

```
1 -- Professores cadastrados
2 • SELECT pro_siape, pro_pnome, pro_unome
3 FROM professor;
```

pro_siape	pro_pnome	pro_unome
1	Hanka	Ruebbert
4	Gerhard	Huettio
5	Angela	Lehmann
6	Lisa	Reimann
7	Corinna	Engellmann
11	Luiz	Marengo
12	Mano	Lima
13	Cenair	Maicá

# Resumo da Aula





# **DISCIPLINA: Banco de Dados 1**

Prof. **GIOVANI** Volnei Meinerz

Aula 20 – SQL (cont.)