

# **Consultas SQL**

## **- SELECT -**

## **PARTE 3**

**Prof. Dr. Anderson Chaves Carniel**

[accarniel@ufscar.br](mailto:accarniel@ufscar.br)



# Base de dados (esquema relacional) considerada

**Empregado** = {PrimeiroNome, InicialMeio, UltimoNome, NumEmpregado, DataNascimento, Endereco, Sexo, Salario, NumSupervisor, NumDept}

**Departamento** = {NomeDept, NumDept, NumGerente, DataInicioGerencia}

**Localizacao\_Dpto** = {NumDept, Localizacao}

**Projeto** = {NomeProj, NumProj, Localizacao, NumDept}

**Trabalha** = {NumEmpregado, NumProj, Horas}

**Dependente** = {NumEmpregado, NomeDependente, Sexo, DataAniversario, Parentesco}

O esquema possui os dados disponibilizados no Moodle  
Carregá-los usando o pgAdmin

# Subconsultas aninhadas

- Subconsulta é uma consulta SQL dentro de outra consulta
  - Ou seja, uma expressão SELECT ... FROM ... WHERE ... Dentro de outra consulta (dentro de uma cláusula SELECT, FROM, ou WHERE)
- Aplicações mais comuns:
  - testes para membros de conjuntos
  - comparações de conjuntos
  - cardinalidade de conjuntos
- A mesma consulta SQL (incluindo as subconsultas) pode ser escrita de diversas formas

# Membros de um conjunto

- O conjunto é uma coleção de valores oriundos de uma subconsulta SQL (SELECT ... FROM ... WHERE)
- Operadores:
  - **IN**
    - Testa se um valor é membro de um conjunto
  - **NOT IN**
    - Verifica a ausência de um valor em um conjunto
- Conjunto pode ser também uma lista de valores informados entre parênteses

## Exemplo operador IN

- Liste os nomes dos empregados cuja data de nascimento coincide com alguma data de início de uma gerência de um departamento

**Como fica em SQL?**

**DICA:** Subconsulta que traz  
as datas de gerência +  
operador IN

## Exemplo operador IN

- Liste os nomes dos empregados cuja data de nascimento coincide com alguma data de início de uma gerência de um departamento

```
SELECT primeironome  
FROM empregado  
WHERE datanascimento IN  
(SELECT datainiciogerencia  
FROM departamento)
```

Subconsulta que vem  
seguida do operador IN

Consulta gera um resultado vazio na base de dados atual disponibilizada

## Exemplo operador NOT IN

- Liste os nomes dos empregados com data de nascimento **diferente** das datas de início de uma gerência de um departamento

**Como fica em SQL?**

**DICA:** Subconsulta que traz  
as datas de gerência +  
operador NOT IN

## Exemplo operador NOT IN

- Liste os nomes dos empregados com data de nascimento diferente das datas de início de uma gerência de um departamento

```
SELECT primeironome  
FROM empregado  
WHERE datanascimento NOT IN  
(SELECT datainiciogerencia  
FROM departamento)
```

Subconsulta que vem seguida  
do operador NOT IN

# Comparações sobre conjuntos

- Existem operadores que podem ser usados sobre conjuntos (como o IN e NOT IN), eles são:
- **SOME (ou ANY)**
  - Retorna *true* para as tuplas que satisfazem algum operador relacional
  - Exemplo:
    - SELECT ... FROM ... WHERE datanascimento > SOME (CONJUNTO SUBSELECT)
    - a condição é verdadeira quando datanascimento for maior que **algum (ou seja, pelo menos 1)** dos resultados presentes na lista (resultado de uma consulta)
  - Uso de operadores relacionais
    - <, <=, >= , >, =, <>

# Comparações sobre conjuntos

- Existem operadores que podem ser usados sobre conjuntos (como o IN e NOT IN), eles são:
- **ALL**
  - Retorna *true* para as tuplas que satisfazem o operador relacional considerando TODAS as tuplas do conjunto
- Exemplo:
  - SELECT ... FROM ... WHERE datanascimento > ALL (CONJUNTO SUBSELECT)
  - a condição é verdadeira quando datanascimento for maior que **todas as datas** presentes na lista (resultado de uma consulta)
- Uso de operadores relacionais
  - <, <=, >= , >, =, <>

# Cardinalidade de conjuntos (divisão relacional)

- Antes de verificar como executar uma divisão relacional usando SQL, é necessário saber as definições dos seguintes operadores:
- **EXISTS**
  - SELECT ... FROM ... WHERE **EXISTS** (lista de valores)
  - a condição é verdadeira quando a lista (resultado de uma sub consulta) **NÃO É VAZIA (OU SEJA, EXISTE ALGUM ELEMENTO)**
  - A quantidade de elementos retornado pela lista é a cardinalidade do conjunto
- **NOT EXISTS**
  - SELECT ... FROM ... WHERE **NOT EXISTS** (lista de valores)
  - a condição é verdadeira quando a lista **É VAZIA**

## Álgebra relacional versus SQL: divisão

- Divisão de duas relações R e S
  - Retorna todos os valores de um atributo de R que fazem referência (ou seja, se associam) a **TODOS OS VALORES** de um atributo de S
- A divisão é utilizada para consultas que incluem o termo **para todos** ou em **todos**

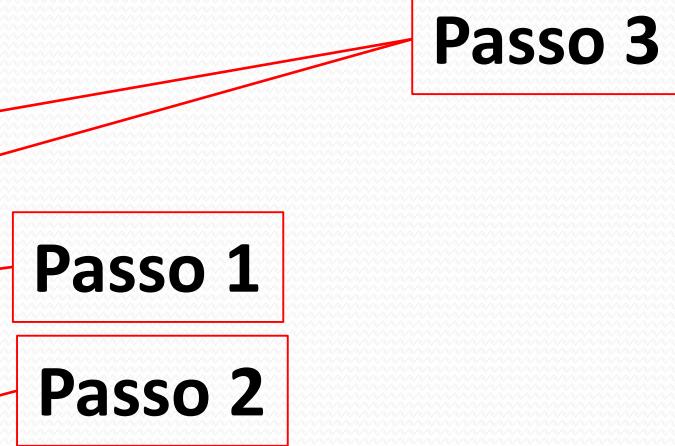
# Álgebra relacional versus SQL: divisão

- Consulta:
  - Lista todos os nomes dos empregados que já trabalharam **em todos** os projetos
  - Consulta em álgebra relacional:
$$\Pi_{\{primeironome, ultimonome, numprojeto\}} \left( empregado^{(numempregado = numempregado)} \bowtie trabalho \right) \\ \div \Pi_{\{numprojeto\}} projeto$$
- Em SQL não existe um operador direto para o operador da álgebra relacional, então é realizado uma sequência de passos
  - Exemplo a seguir mostra esses passos aplicados a essa consulta (o raciocínio da divisão também se aplica a execução da álgebra relacional)

# Álgebra relacional versus SQL: divisão

- Consulta SQL:

```
SELECT primeironome, ultimonome
FROM empregado
WHERE NOT EXISTS
( (SELECT DISTINCT numproj
  FROM trabalha)
EXCEPT
(SELECT numproj
  FROM trabalha
  WHERE empregado.numempregado = trabalha.numempregado)
)
```



Para cada tupla de empregado (especificado fora desse subselect), será executado o WHERE

# Álgebra relacional versus SQL: divisão

- **Passo 1:**

- Capturar todos os elementos que queremos corresponder
- No exemplo, os números do projeto que alguém está trabalhando ou trabalhou!
- Consulta:  
SELECT DISTINCT numproj FROM trabalha

	<b>numempregado [PK] integer</b>	<b>numproj [PK] integer</b>	<b>horas integer</b>
<b>1</b>	<b>1</b>	<b>1</b>	<b>10</b>
<b>2</b>	<b>1</b>	<b>2</b>	<b>10</b>
<b>3</b>	<b>1</b>	<b>3</b>	<b>10</b>
<b>4</b>	<b>3</b>	<b>2</b>	<b>40</b>
<b>5</b>	<b>4</b>	<b>1</b>	<b>100</b>
<b>6</b>	<b>5</b>	<b>1</b>	<b>40</b>
<b>7</b>	<b>5</b>	<b>2</b>	<b>100</b>
<b>8</b>	<b>5</b>	<b>3</b>	<b>10</b>
<b>9</b>	<b>7</b>	<b>1</b>	<b>300</b>
<b>10</b>	<b>11</b>	<b>3</b>	<b>390</b>

# Álgebra relacional versus SQL: divisão

- **Passo 2:**

- Capturar todos os elementos que já contém a correspondência, ou seja, fazendo a junção com os elementos da primeira relação (externa)

- No exemplo, os projetos de um empregado específico!
- Consulta:

```
SELECT numproj
```

```
FROM trabalha
```

```
WHERE empregado.numempregado =
trabalha.numempregado
```

**Cadê o empregado na cláusula FROM? Ele estará na consulta de fora!**

Então, para cada empregado especificado no lado de fora dessa consulta, é feito uma junção

Empregado		Trabalha		Resultado	
		numempregado [PK] integer	numproj [PK] integer	horas integer	numproj integer
	1	1	1	10	1
	2	1	2	10	3
	3	1	3	10	2
	4	3	2	40	
	5	4	1	100	
	6	5	1	40	
	7	5	2	100	
	8	5	3	10	
	9	7	1	300	
	10	11	3	390	

Descobrindo os projetos que o empregado 1 trabalha

# Álgebra relacional versus SQL: divisão

- **Passo 3:**
  - Aplicar a diferença (EXCEPT) e o NOT EXISTS
  - Uma vez que sabemos todos os projetos que possuem trabalhadores (passo 1) e todos os projetos que um determinado empregado trabalha (passo 2), temos que descobrir se existe alguma tupla resultante da diferença entre essas duas tabelas
    - **Se existir uma ou mais tuplas:** a diferença nos indica que tem algum projeto que o empregado não trabalha
    - **Se não existir tupla:** a diferença nos indica que o empregado trabalha em todos os projetos
    - Queremos esse caso! Então usa-se o **NOT EXISTS**

	<b>primeironome</b> character varying(50)	<b>ultimonome</b> character varying(50)
<b>1</b>	Ana	Silva
<b>2</b>	André	Rasp

## Divisão – exercício

- **Listar todos os empregados que trabalharam em todos os projetos dos quais o empregado chamado Lucia participou**

# Divisão – resposta

- Listar todos os empregados que trabalharam em todos os projetos dos quais o empregado chamado Lucia participou

```
SELECT primeironome, ultimonome
FROM empregado
WHERE NOT EXISTS
```

```
( (SELECT DISTINCT numproj
  FROM trabalha
  WHERE trabalha.numempregado IN (SELECT numempregado FROM empregado WHERE
primeironome = 'Lucia'))
```

```
EXCEPT
(SELECT numproj
  FROM trabalha
  WHERE empregado.numempregado = trabalha.numempregado)
)
```

```
AND primeironome <> 'Lucia'
```

**aqui restringiu os nossos elementos de interesse, somente para os projetos da Lucia**

**colocou essa condição para remover o nome da Lucia no resultado final, pois claramente ela participa de todos os projetos dela mesmo**