

# **Consultas SQL**

## **- SELECT -**

## **PARTE 1**

**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}

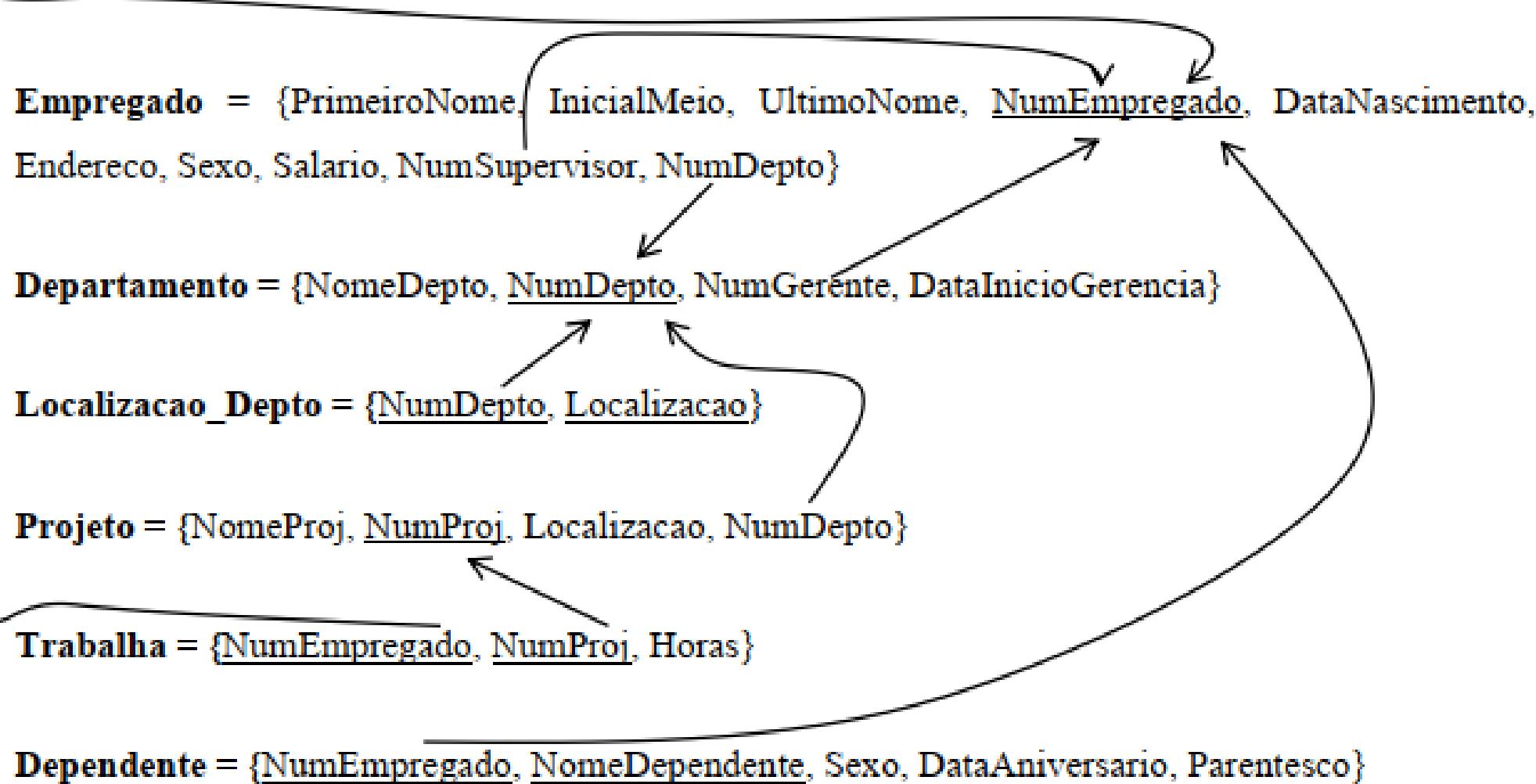
O esquema possui os dados disponibilizados no Moodle  
Carregá-los usando o pgAdmin

# Álgebra relacional VERSUS SQL

- Condições no WHERE correspondem ao operador de **seleção**
- Deseja-se recuperar os dependentes do sexo M**

numempregado integer	nomedependente character varying(150)	sexo character(1)	dataaniversario date	parentesco character varying(100)
1	Nicoli	F	2014-01-29	filho
1	Pita	F	2015-01-29	filho
1	Raissa	F	2016-01-29	filho
9	Rodrigo	M	2014-01-29	filho
9	Roberto	M	2019-01-29	filho
2	Roberta	F	2000-01-29	conjugue
5	Nicolai	M	2018-01-29	filho
4	Vicente		1963-01-29	conjugue

- Como fica em álgebra relacional?
- **Deseja-se recuperar os dependentes do sexo M**



# Álgebra relacional VERSUS SQL

- Álgebra relacional:  $\sigma_{(sexo = 'M')} dependente$

# Álgebra relacional VERSUS SQL

- Álgebra relacional:  $\sigma_{(sexo = 'M')} dependente$
- Consulta SQL: SELECT \* FROM dependente WHERE sexo = 'M'

numempregado integer	nomedependente character varying(150)	sexo character(1)	dataaniversario date	parentesco character varying(100)
9	Rodrigo	M	2014-01-29	filho
9	Roberto	M	2019-01-29	filho
5	Nicolai	M	2018-01-29	filho

# Álgebra relacional VERSUS SQL

- Colunas na cláusula SELECT correspondem ao operador de **projeção**
- Deseja-se somente recuperar o **nomedependente** e **dataaniversario**

numempregado integer	nomedependente character varying(150)	sexo character(1)	dataaniversario date	parentesco character varying(100)
1	Nicoli	F	2014-01-29	filho
1	Pita	F	2015-01-29	filho
1	Raissa	F	2016-01-29	filho
9	Rodrigo	M	2014-01-29	filho
9	Roberto	M	2019-01-29	filho
2	Roberta	F	2000-01-29	conjugue
5	Nicolai	M	2018-01-29	filho
4	Vicente		1963-01-29	conjugue

# Álgebra relacional VERSUS SQL

- Colunas na cláusula SELECT correspondem ao operador de **projeção**
  - Deseja-se somente recuperar o **nomedependente** e **dataaniversario**

numempregado integer	nomedependente character varying(150)	sexo character(1)	dataaniversario date	parentesco character varying(100)
1	Nicoli	F	2014-01-29	filho
1	Pita	F	2015-01-29	filho
9	Roberto	M	2019-01-29	filho
2	Roberta	F	2000-01-29	conjugue
5	Nicolai	M	2018-01-29	filho
4	Vicente		1963-01-29	conjugue

**COMO (em álgebra relacional)??**

# Álgebra relacional VERSUS SQL

- Álgebra relacional:

$$\Pi_{\{nomedependente, dataaniversario\}} \sigma_{(sexo = 'M')} dependente$$

- Consulta SQL:

```
SELECT nomedependente, dataaniversario  
FROM dependente WHERE sexo = 'M'
```

nomedependente character varying(150)	dataaniversario date
Rodrigo	2014-01-29
Roberto	2019-01-29
Nicolai	2018-01-29

# Álgebra relacional VERSUS SQL

- Tabelas na cláusula FROM correspondem ao uso do operador **Produto Cartesiano**

nomeproj character varying(100)	numproj integer	localizacao character varying(100)	numdepto integer
proposta do produto X	1	sala 54	1
treinamento de pessoal	2	sala 4	3
melhorias no financeiro	3	sala 4	3

**Tabela projeto**

nomedepo character varying(50)	numdepto integer	numgerente integer	datainiciogerencia date
Diretoria	1	1	2010-10-10
Compras	2	5	2010-10-10
Vendas	3	3	2010-10-10

**Tabela departamento**

# Álgebra relacional VERSUS SQL

- Tabelas na cláusula FROM correspondem ao uso do operador **Produto Cartesiano**

nomeproj character varying(100)	numproj integer	localizacao character varying(100)	numdepto integer
proposta do produto X	1	sala 54	1
treinamento de pessoal	2	sala 4	3
melhorias no financeiro	3	sala 4	3

**Tabela projeto**

**COMO fica (em álgebra relacional)??**

nomedeppto character varying(50)	numdepto integer	numgerente integer	datainiciogerencia date
Diretoria	1	1	2010-10-10
Compras	2	5	2010-10-10
Vendas	3	3	2010-10-10

**Tabela departamento**

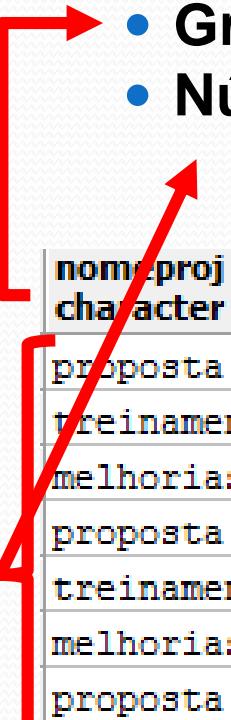
# Álgebra relacional VERSUS SQL

- Álgebra relacional: *departamento* × *projeto*
- Consulta SQL:  
SELECT \* FROM departamento, projeto

nomeproj character varying(100)	numproj integer	localizacao character varying(100)	numdepto integer	nomedeppto character varying(50)	numdepto integer	numgerente integer	datainiciogerencia date
proposta do produto X	1	sala 54	1	Diretoria	1	1	2010-10-10
treinamento de pessoal	2	sala 4	3	Diretoria	1	1	2010-10-10
melhorias no financeiro	3	sala 4	3	Diretoria	1	1	2010-10-10
proposta do produto X	1	sala 54	1	Compras	2	5	2010-10-10
treinamento de pessoal	2	sala 4	3	Compras	2	5	2010-10-10
melhorias no financeiro	3	sala 4	3	Compras	2	5	2010-10-10
proposta do produto X	1	sala 54	1	Vendas	3	3	2010-10-10
treinamento de pessoal	2	sala 4	3	Vendas	3	3	2010-10-10
melhorias no financeiro	3	sala 4	3	Vendas	3	3	2010-10-10

# Considerações sobre produto cartesiano

- Operação que combina tuplas de 2 ou mais relações
- Tuplas da tabela resultante:
  - Todas as combinações possíveis
- **Grau:** soma do número de atributos de ambas as relações
- **Número de tuplas:** resultado da multiplicação da quantidade de tuplas das relações



nomeproj character varying(100)	numproj integer	localizacao character varying(100)	numdepto integer	nomedepo character varying(50)	numdepto integer	numgerente integer	datainicio date
proposta do produto X	1	sala 54		Diretoria	1	1	2010-10-10
treinamento de pessoal	2	sala 4		Diretoria	1	1	2010-10-10
melhorias no financeiro	3	sala 4		Diretoria	1	1	2010-10-10
proposta do produto X	1	sala 54		Compras	2	5	2010-10-10
treinamento de pessoal	2	sala 4		Compras	2	5	2010-10-10
melhorias no financeiro	3	sala 4		Compras	2	5	2010-10-10
proposta do produto X	1	sala 54		Vendas	3	3	2010-10-10
treinamento de pessoal	2	sala 4		Vendas	3	3	2010-10-10
melhorias no financeiro	3	sala 4		Vendas	3	3	2010-10-10

# Álgebra relacional VERSUS SQL

Álgebra relacional	SQL
Projeção	SELECT
Produto cartesiano	FROM
Seleção	WHERE

# Álgebra relacional VERSUS SQL

## Junção

- concatenar tuplas **relacionadas** de duas relações
- Passos:
  - formar um **produto cartesiano** das relações (**listar relações no FROM**)
  - fazer uma **seleção** forçando igualdade sobre os atributos que aparecem nas relações (**colocar condição de junção no WHERE**)

## Exemplo: Junção

**COMO fica (em álgebra relacional e em SQL)??**

- Exemplo: selecionar os projetos de cada departamento
- **Ideia: relacionar os projetos com os departamentos**



nomeproj character varying(100)	numproj integer	localizacao character varying(100)	numdepto integer	nomedeppto character varying(50)	numdepto integer	numgerente integer	datainiciogerencia date
proposta do produto X	1	sala 54	1	Diretoria	1	1	2010-10-10
treinamento de pessoal	2	sala 4	3	Diretoria	1	1	2010-10-10
melhorias no financeiro	3	sala 4	3	Diretoria	1	1	2010-10-10
proposta do produto X	1	sala 54	1	Compras	2	5	2010-10-10
treinamento de pessoal	2	sala 4	3	Compras	2	5	2010-10-10
melhorias no financeiro	3	sala 4	3	Compras	2	5	2010-10-10
proposta do produto X	1	sala 54	1	Vendas	3	3	2010-10-10
treinamento de pessoal	2	sala 4	3	Vendas	3	3	2010-10-10
melhorias no financeiro	3	sala 4	3	Vendas	3	3	2010-10-10

# Álgebra relacional VERSUS SQL

- Álgebra relacional:  $departamento \underset{\text{numdepto} = \text{numdepto}}{\bowtie} projeto$
- Consulta SQL:
 

```
SELECT * FROM departamento as d, projeto as p
WHERE d.numdepto = p.numdepto
```

nomedeppto character varying(50)	numdepto integer	numgerente integer	datainiciogerencia date	nomeproj character varying(100)	numproj integer	localizacao character varying(100)	numdepto integer
Diretoria	1	1	2010-10-10	proposta do produto X	1	sala 54	1
Vendas	3	3	2010-10-10	melhorias no financeiro	3	sala 4	3
Vendas	3	3	2010-10-10	treinamento de pessoal	2	sala 4	3

# Álgebra relacional VERSUS SQL

- Álgebra relacional:  $departamento \underset{\text{numdepto}=\text{numdepto}}{\bowtie} projeto$
- Consulta SQL:

```
SELECT * FROM departamento as d, projeto as p
WHERE d.numdepto = p.numdepto
```

O nome de departamento é usado aqui

Uso de apelido para renomear as tabelas

nomedep character varying(50)	numdepto integer	numgerente integer	datainiciogerencia date	nomeproj character varying(100)	numproj integer	localizacao character varying(100)	numdepto integer
Diretoria	1	1	2010-10-10	proposta do produto X	1	sala 54	1
Vendas	3	3	2010-10-10	melhorias no financeiro	3	sala 4	3
Vendas	3	3	2010-10-10	treinamento de pessoal	2	sala 4	3

# Álgebra relacional VERSUS SQL combinando operadores

- Consulta que retorna o nome dos projetos mantidos pelo departamento Diretoria
- Álgebra relacional:

???

- Consulta SQL:

???

**COMO fica (em álgebra relacional e em SQL)??**

# Álgebra relacional VERSUS SQL combinando operadores

- Consulta que retorna o nome dos projetos mantidos pelo departamento Diretoria

- Álgebra relacional:

$$\Pi_{\{nomeproj\}} (\sigma_{(nomedepo='Diretoria')} departamento)^{(numdepto=numdepto)} \bowtie projeto$$

- Consulta SQL:

```
SELECT nomeproj FROM departamento as d, projeto as p
WHERE d.numdepto = p.numdepto AND nomedepo = 'Diretoria'
```

nomeproj
character varying(100)
proposta do produto X

# Álgebra relacional VERSUS SQL combinando operadores

- Consulta que retorna o nome dos projetos mantidos pelo departamento Diretoria

- Álgebra relacional:

$$\Pi_{\{nomeproj\}} (\sigma_{(nomedepo='Diretoria')} departamento)^{(numdepto=numdepto)} \bowtie projeto$$

- Consulta SQL:

```
SELECT nomeproj as Projeto FROM departamento as d, projeto as p
WHERE d.numdepto = p.numdepto AND nomedepo = 'Diretoria'
```

Uso de apelido para renomear coluna

<b>projeto</b>
<b>character varying(100)</b>

<b>proposta do produto X</b>
------------------------------

# Álgebra relacional VERSUS SQL: Joins

- **[INNER] JOIN**
  - Operação vista anteriormente – somente mantém as tuplas com correspondência em ambas as relações
- Em SQL, pode-se também especificar a junção no FROM usando o operador **JOIN**
  - O operador **JOIN** é principalmente usado para os casos de junção externa:
    - **LEFT [OUTER] JOIN**
    - **RIGHT [OUTER] JOIN**
    - **FULL [OUTER] JOIN**

Elementos entre colchetes são opcionais na escrita da consulta (ou seja, não precisam ser especificados)

# Álgebra relacional VERSUS SQL: Inner Join

- [INNER] JOIN
  - Todos os nomes de departamentos e seus projetos

$\Pi_{\{nomedepo, nomeproj\}} departamento^{(numdepto = numdepto)} \bowtie projeto$

**Equivalente à**

SELECT nomedepo, nomeproj FROM departamento as d, projeto as p  
WHERE d.numdepto = p.numdepto

**Equivalente à**

SELECT nomedepo, nomeproj FROM departamento as d **JOIN** projeto as p **ON**  
d.numdepto = p.numdepto

nomedepo character varying(50)	nomeproj character varying(100)
Diretoria	proposta do produto X
Vendas	melhorias no financeiro
Vendas	treinamento de pessoal

# Álgebra relacional VERSUS SQL: Left outer Join

- **LEFT [OUTER] JOIN**

- mantém cada tupla da relação à **esquerda** na tabela de junção e preenche com valores nulos as tuplas da tabela à **direita** da junção que não correspondem à condição de junção
- Todos os nomes de departamentos e seus projetos, mesmo aqueles departamentos sem projeto:

**COMO fica (em álgebra relacional e em SQL)??**

# Álgebra relacional VERSUS SQL: Left outer Join

- **LEFT [OUTER] JOIN**

- mantém cada tupla da relação à **esquerda** na tabela de junção e preenche com valores nulos as tuplas da tabela à **direita** da junção que não correspondem à condição de junção
- Todos os nomes de departamentos e seus projetos, mesmo aqueles departamentos sem projeto:

$$\Pi_{\{nomedepo, nomeproj\}} \text{departamento}^{(numdepto = numdepto)} \text{projeto}$$


**Equivalente à**

SELECT nomedepo, nomeproj FROM departamento as d **LEFT JOIN** projeto as p **ON**  
d.numdepto = p.numdepto

nomedepo character varying(50)	nomeproj character varying(100)
Diretoria	proposta do produto X
Compras	
Vendas	melhorias no financeiro
Vendas	treinamento de pessoal

# Álgebra relacional VERSUS SQL: Right outer Join

$\Pi_{\{nomedependente, primeironome\}} dependente \bowtie^{(numempregado = numempregado)} empregado$

- **RIGHT [OUTER] JOIN**

- mantém cada tupla da relação à **direita** na tabela de junção e preenche com valores nulos as tuplas da tabela à **esquerda** da junção que não correspondem à condição de junção
- Todos os nomes de empregados e seus dependentes, mesmo os empregados sem dependentes:

A álgebra relacional é equivalente a:

???????

nomedependente character varying(150)	primeironome character varying(50)
Raissa	Ana
Pita	Ana
Nicoli	Ana
Roberta	Romeu
Vicente	Lucia
	Fran
	Antonio
	Geraldo
Nicolai	André
Roberto	Carlos
Rodrigo	Carlos
	Carla
	Karla
	Julietta

# Álgebra relacional VERSUS SQL: Right outer Join

$\Pi_{\{nomedependente, primeironome\}} dependente^{(numempregado = numempregado)} \bowtie empregado$

- **RIGHT [OUTER] JOIN**

- mantém cada tupla da relação à **direita** na tabela de junção e preenche com valores nulos as tuplas da tabela à **esquerda** da junção que não correspondem à condição de junção
- Todos os nomes de empregados e seus dependentes, mesmo os empregados sem dependentes:

A álgebra relacional é equivalente a:

SELECT nomedependente, primeironome FROM dependente as d **RIGHT JOIN** empregado as e ON d.numempregado = e.numempregado

nomedependente character varying(150)	primeironome character varying(50)
Raissa	Ana
Pita	Ana
Nicoli	Ana
Roberta	Romeu
Vicente	Lucia
	Fran
	Antonio
	Geraldo
Nicolai	André
Roberto	Carlos
Rodrigo	Carlos
	Carla
	Karla
	Julietta

# Álgebra relacional VERSUS SQL: Full outer Join

- **FULL [OUTER] JOIN**

- mantém todas as tuplas das duas relações, correlacionando as comuns e colocando valor *null* nas tuplas que não correspondem à condição de junção

$R \bowtie_{A=A} S$

R			S		$R \bowtie_{A=A} S$				
A	B	C	A	D	R.A	S.A	B	C	D
1	a	x	1	d	1	1	a	x	d
2	b	y	2	d	2	2	b	y	d
3	a	y	5	e	3	Null	a	y	Null
4	c	y			4	Null	c	y	Null
					Null	5	Null	Null	e

A álgebra relacional é equivalente a:

```
SELECT r.a, s.a, b, c, d FROM R
FULL JOIN S ON
R.A = S.A
```

# Álgebra relacional VERSUS SQL

Álgebra relacional	SQL
$\bowtie$	JOIN (ou produto cartesiano + condições de junção no WHERE)
$\bowtie_L$	LEFT JOIN
$\bowtie_R$	RIGHT JOIN
$\bowtie_{LR}$	FULL JOIN

# ORDER BY SQL

- Ordena as tuplas que aparecem no resultado de uma consulta
- Tipos de ordenação:
  - **ASC** (é o padrão): ordem ascendente
  - **DESC**: ordem descendente
- Ordenação pode ser especificada em vários atributos
  - Neste caso, a ordenação referente ao primeiro atributo é prioritária. Se houver valores repetidos, então é utilizada a ordenação referente ao segundo atributo, e assim por diante

## Exemplo ORDER BY

```
SELECT nomedependente, dataaniversario, parentesco  
FROM dependente  
--condições podem ser colocados no WHERE  
ORDER BY parentesco DESC, nomedependente
```

nomedependente character varying(150)	dataaniversario date	parentesco character varying(100)
Nicolai	2018-01-29	filho
Nicoli	2014-01-29	filho
Pita	2015-01-29	filho
Raissa	2016-01-29	filho
Roberto	2019-01-29	filho
Rodrigo	2014-01-29	filho
Roberta	2000-01-29	conjugue
Vicente	1963-01-29	conjugue

Nesta consulta, ordena-se primeiro, em ordem descendente, o *parentesco* e depois, em ordem ascendente, os *nome dos dependentes*