

Sistemas de Informação
GSI016 Banco de Dados 1

SQL

Linguagem de Manipulação de Dados

Profa. Maria Camila Nardini Barioni
camila.barioni@facom.ufu.br

Bloco B - sala 1B137

1º semestre de 2020

Discussão exercício

◆ Modelo Relacional

Aluno = { NumAluno, Nome, Endereco, Cidade, Telefone,
NumCurso(Curso.NumCurso) }

Disciplina = { NumDisp, Nome, QuantCreditos }

Professor = { NumFunc, Nome, Admissao, AreaPesquisa }

Curso = { NumCurso, Nome, TotalCréditos }

Aula = { NumAluno(Aluno.NumAluno),
NumDisp(Disciplina.NumDisp),
NumFunc(Professor.NumFunc), Semestre, Nota }

DisciplinaCurso = { NumDisp(Disciplina.NumDisp),
NumCurso(Curso.NumCurso) }

Discussão exercício

/ e) Quais os nomes das disciplinas do curso de Ciência da Computação. */*

Disciplina = { NumDisp, Nome, QuantCreditos }

Curso = { NumCurso, Nome, TotalCréditos }

DisciplinaCurso = { NumDisp(Disciplina.NumDisp),
NumCurso(Curso.NumCurso) }

Discussão exercício

/ e) Quais os nomes das disciplinas do curso de Ciência da Computação. */*

```
SELECT D.nome
FROM Disciplina D, Curso C, DisciplinaCurso DC
WHERE D.numdisp = DC.numdisp AND
        DC.numcurso = C.numcurso AND
        C.nome = 'Ciência da Computação'
```

Discussão exercício

/ e) Quais os nomes das disciplinas do curso de Ciência da Computação. */*

```
SELECT D.nome
FROM Disciplina D, Curso C, DisciplinaCurso DC
WHERE D.numdisp = DC.numdisp AND
      DC.numcurso = C.numcurso AND
      C.nome = 'Ciência da Computação'
```



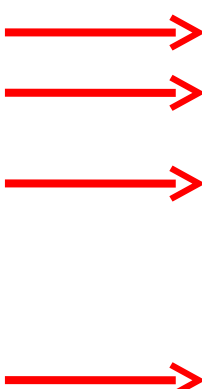
Curso		
<u>NumCurs</u> o	Nome	TotalCrédit os
2142	Engenharia Civil	1500
2143	Ciência da Computação	2000
2144	Direito	1750
2145	Pedagogia	1500
2146	Odontologia	1600

Discussão exercício

/ e) Quais os nomes das disciplinas do curso de Ciência da Computação. */*

```
SELECT D.nome
FROM Disciplina D, Curso C, DisciplinaCurso DC
WHERE D.numdisp = DC.numdisp AND
      DC.numcurso = C.numcurso AND
      C.nome = 'Ciência da Computação'
```

2143



DisciplinaCurso	
<u>NumDisp</u>	<u>NumCurso</u>
1	2143
2	2143
3	2144
4	2143
4	2142
5	2145
6	2144
7	2143
7	2142

Discussão exercício

/ e) Quais os nomes das disciplinas do curso de Ciência da Computação. */*

```
SELECT D.nome
FROM Disciplina D, Curso C, DisciplinaCurso DC
WHERE D.numdisp = DC.numdisp AND
      DC.numcurso = C.numcurso AND
      C.nome = 'Ciência da Computação'
```

1, 2, 4 e 7

Disciplina		
NumDisp	Nome	QuantCreditos
1	Banco de Dados	30
2	Estrutura de Dados	30
3	Direito Penal	25
4	Cálculo Numérico	30
5	Psicologia Infantil	25
6	Direito Tributário	33
7	Engenharia Software	27

Discussão

/* e) Quais os

SELECT D.no

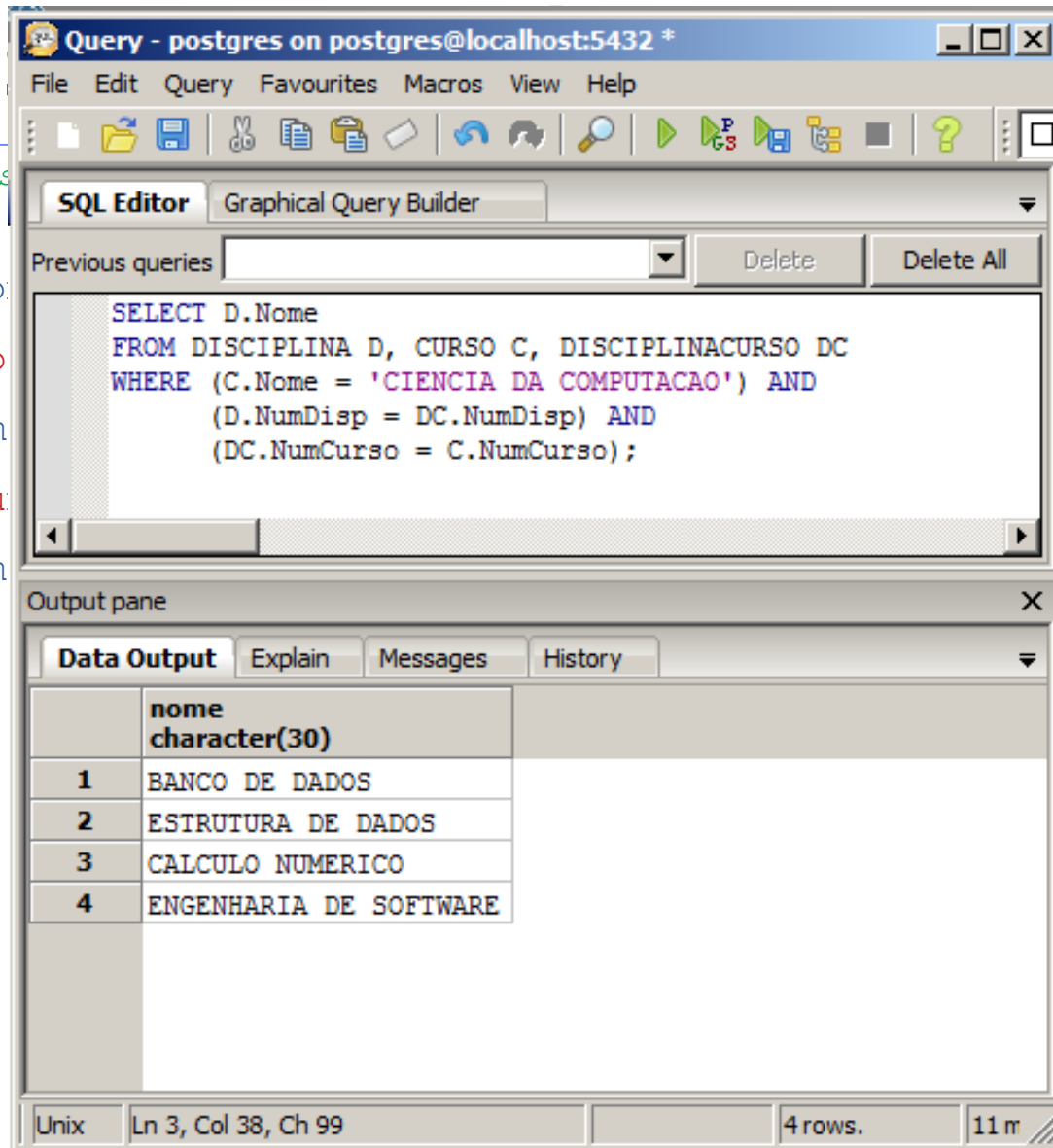
FROM Discip

WHERE D.num

DC.nu

C.nom

ação. */



Discussão exercício

```
/* m) Quais nomes de alunos, nomes de disciplinas e  
notas do 1º semestre de 1998 no curso de Ciência da  
Computação.*/
```

```
Aluno          = { NumAluno, Nome, Endereco, Cidade, Telefone,  
                  NumCurso(Curso.NumCurso) }
```

```
Disciplina     = { NumDisp, Nome, QuantCreditos }
```

```
Curso          = { NumCurso, Nome, TotalCréditos }
```

```
Aula           = { NumAluno(Aluno.NumAluno),  
                  NumDisp(Disciplina.NumDisp),  
                  NumFunc(Professor.NumFunc), Semestre, Nota }
```

Discussão exercício

/ m) Quais nomes de alunos, nomes de disciplinas e notas do 1º semestre de 1998 no curso de Ciência da Computação.*/*

```
SELECT A.nome, D.nome, AU.nota
FROM Aluno A, Disciplina D, Aula AU, Curso C
WHERE A.numcurso = C.numcurso AND
      C.nome = 'Ciência da Computação' AND -- Alunos do curso CC
      A.numaluno = AU.numaluno AND
      AU.semestre = '01/1998' AND          -- Filtrando o semestre
      AU.numdisp = D.numdisp
```



SQL DML – CONTINUAÇÃO...

SQL DML

◆ SELECT ... FROM ... WHERE ...

- lista atributos de uma ou mais tabelas de acordo com alguma condição

◆ INSERT INTO ...

- insere dados em uma tabela

◆ DELETE FROM ... WHERE ...

- remove dados de tabelas já existentes

◆ UPDATE ... SET ... WHERE ...

- altera dados específicos de uma tabela

SQL DML

◆ SELECT ... FROM ... WHERE ...

- lista atributos de uma ou mais tabelas de acordo com alguma condição

◆ INSERT INTO ...

- insere dados em uma tabela

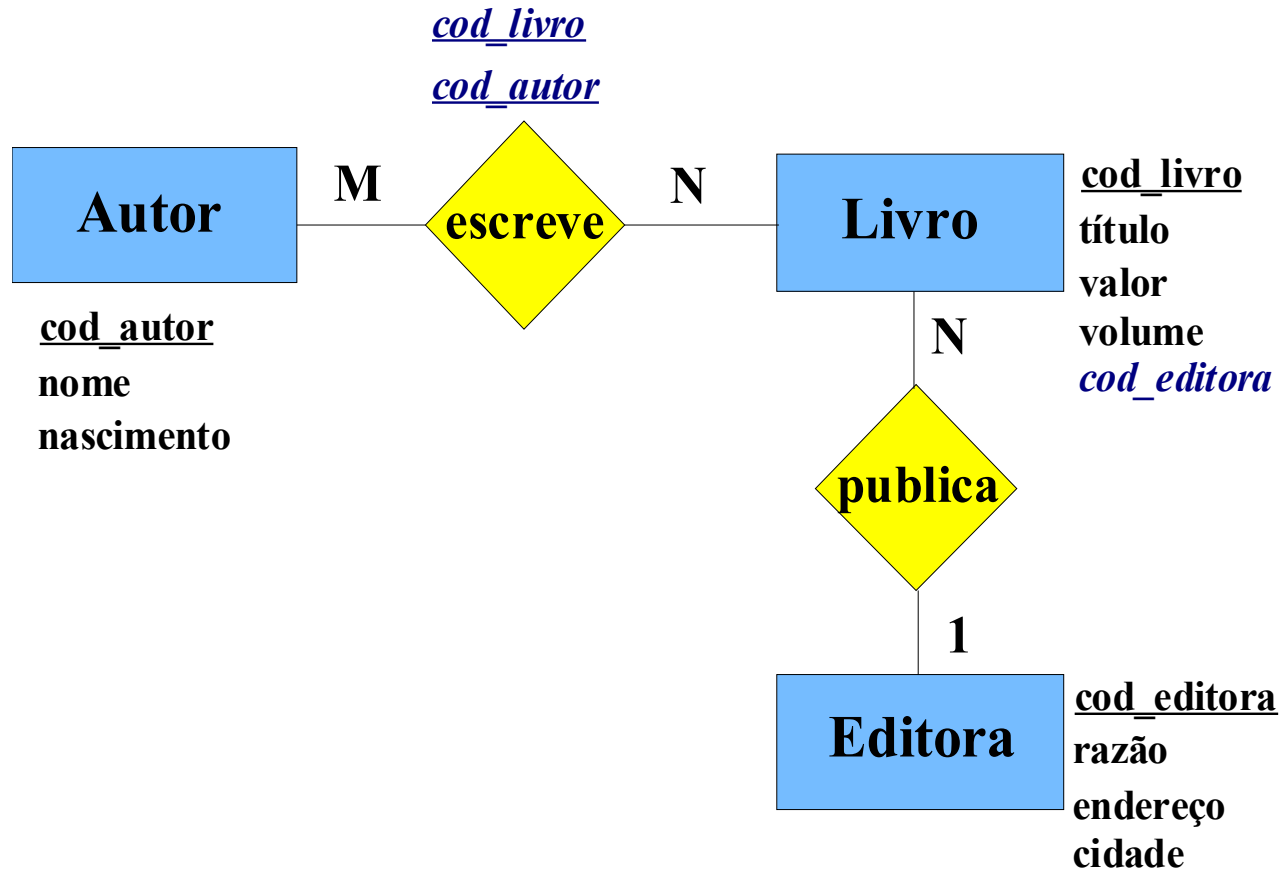
◆ DELETE FROM ... WHERE ...

- remove dados de tabelas já existentes

◆ UPDATE ... SET ... WHERE ...

- altera dados específicos de uma tabela

Exemplo ME-R



Exemplo Modelo Relacional

AUTOR = { COD_AUTOR, NOME, NASCIMENTO }



ESCREVE = { COD_AUTOR, COD_LIVRO }



LIVRO = { COD_LIVRO, TITULO, VALOR,
VOLUME, COD_EDITORA }



EDITORA = { COD_EDITORA, RAZAO, ENDERECO,
CIDADE }

Funções de Agregação

Function	Argument Type(s)	Return Type	Description
<code>array_agg(expression)</code>	any	array of the argument type	input values, including nulls, concatenated into an array
<code>avg(expression)</code>	smallint, int, bigint, real, double precision, numeric, or interval	numeric for any integer-type argument, double precision for a floating-point argument, otherwise the same as the argument data type	the average (arithmetic mean) of all input values
<code>bit_and(expression)</code>	smallint, int, bigint, or bit	same as argument data type	the bitwise AND of all non-null input values, or null if none
<code>bit_or(expression)</code>	smallint, int, bigint, or bit	same as argument data type	the bitwise OR of all non-null input values, or null if none
<code>bool_and(expression)</code>	bool	bool	true if all input values are true, otherwise false
<code>bool_or(expression)</code>	bool	bool	true if at least one input value is true, otherwise false
<code>count(*)</code>		bigint	number of input rows
<code>count(expression)</code>	any	bigint	number of input rows for which the value of <i>expression</i> is not null
<code>every(expression)</code>	bool	bool	equivalent to <code>bool_and</code>
<code>max(expression)</code>	any array, numeric, string, or date/time type	same as argument type	maximum value of <i>expression</i> across all input values
<code>min(expression)</code>	any array, numeric, string, or date/time type	same as argument type	minimum value of <i>expression</i> across all input values
<code>string_agg(expression, delimiter)</code>	(text, text) or (bytea, bytea)	same as argument types	input values concatenated into a string, separated by delimiter
<code>sum(expression)</code>	smallint, int, bigint, real, double precision, numeric, or interval	bigint for smallint or int arguments, numeric for bigint arguments, double precision for floating-point arguments, otherwise the same as the argument data type	sum of <i>expression</i> across all input values
<code>xmlagg(expression)</code>	xml	xml	concatenation of XML values (see also Section 9.14.1.7)

<http://www.postgresql.org/docs/9.2/static/functions-aggregate.html>

Funções de Agregação

Detalhamento e exemplos

- ◆ **MIN()** : Retorna o mínimo de um conjunto de valores contido em um campo especificado em uma consulta.
- ◆ **MAX()** : Retorna o máximo de um conjunto de valores contido em um campo especificado em uma consulta.

-- Verificar o valor do livro mais caro

```
SELECT MAX (VALOR)  
FROM LIVRO;
```

-- E se quisermos saber o título do livro mais caro?

Funções de Agregação

Detalhamento e exemplos

-- E se quisermos saber o título do livro mais caro?
-- Alguns alunos poderiam ter pensado na seguinte
-- solução

```
SELECT TITULO FROM LIVRO WHERE VALOR = MAX(VALOR) ;  
-- Funciona?
```

Funções de Agregação

Detalhamento e exemplos

```
-- E se quisermos saber o título do livro mais caro?  
-- Alguns alunos poderiam ter pensado na seguinte  
-- solução
```

```
SELECT TITULO FROM LIVRO WHERE VALOR = MAX(VALOR);  
-- Funciona?  
-- A função MAX não pode ser usada na cláusula WHERE
```

```
-- e agora?
```

```
SELECT TITULO, VALOR  
  FROM LIVRO  
 WHERE VALOR = (SELECT MAX(VALOR) FROM LIVRO);  
-- OK!
```

Funções de Agregação

Detalhamento e exemplos

◆ **SUM()**: Retorna a soma de um conjunto de valores contido em um campo especificado em uma consulta. A função SUM ignora os registros que contenham campos Null

```
-- calcular o preço total dos livros de Paulo Coelho
```

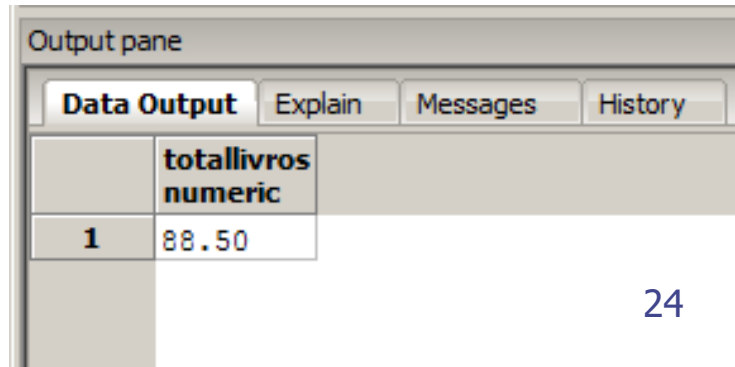
Funções de Agregação

Detalhamento e exemplos

◆ **SUM()**: Retorna a soma de um conjunto de valores contido em um campo especificado em uma consulta. A função `SUM` ignora os registros que contenham campos Null

-- calcular o preço total dos livros de Paulo Coelho

```
SELECT SUM(V valor) AS TotalLivros
FROM LIVRO, AUTOR, ESCREVE
WHERE NOME = 'PAULO COELHO' AND
      COD_AUTOR = COD_AUTOR_ESC AND
      COD_LIVRO_ESC = COD_LIVRO;
```



The screenshot shows a window titled "Output pane" with tabs for "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is active, displaying a table with two columns: "totallivros" and "numeric". The table contains one row with the value "88.50".

	totallivros numeric
1	88.50

Funções de Agregação

Detalhamento e exemplos

◆ **AVG()**: Calcula a média aritmética de um conjunto de valores contido em um campo especificado em uma consulta

-- calcular o preço médio dos livros da livraria

```
SELECT AVG (VALOR) AS Media  
FROM LIVRO;
```

Funções de Agregação

Detalhamento e exemplos

◆ **COUNT()**: Calcula o número de registros retornado por uma consulta. A função COUNT não conta registros que tenham campos NULL, exceto quando expr for o caractere curinga asterisco (*)

-- contar quantos autores estão cadastrados

```
SELECT COUNT(*) FROM AUTOR;
```

-- contar quantos livros possuem informação de volume

```
SELECT COUNT(VOLUME)  
FROM LIVRO;
```

-- observe que registros nos quais VOLUME é NULL não são considerados

Funções de Agregação

Detalhamento e exemplos

◆ COUNT()

- é possível usar a palavra chave DISTINCT. Desta forma, os valores duplicados nas colunas não são contados.

/* qual o número de autores que possuem livros na livraria */

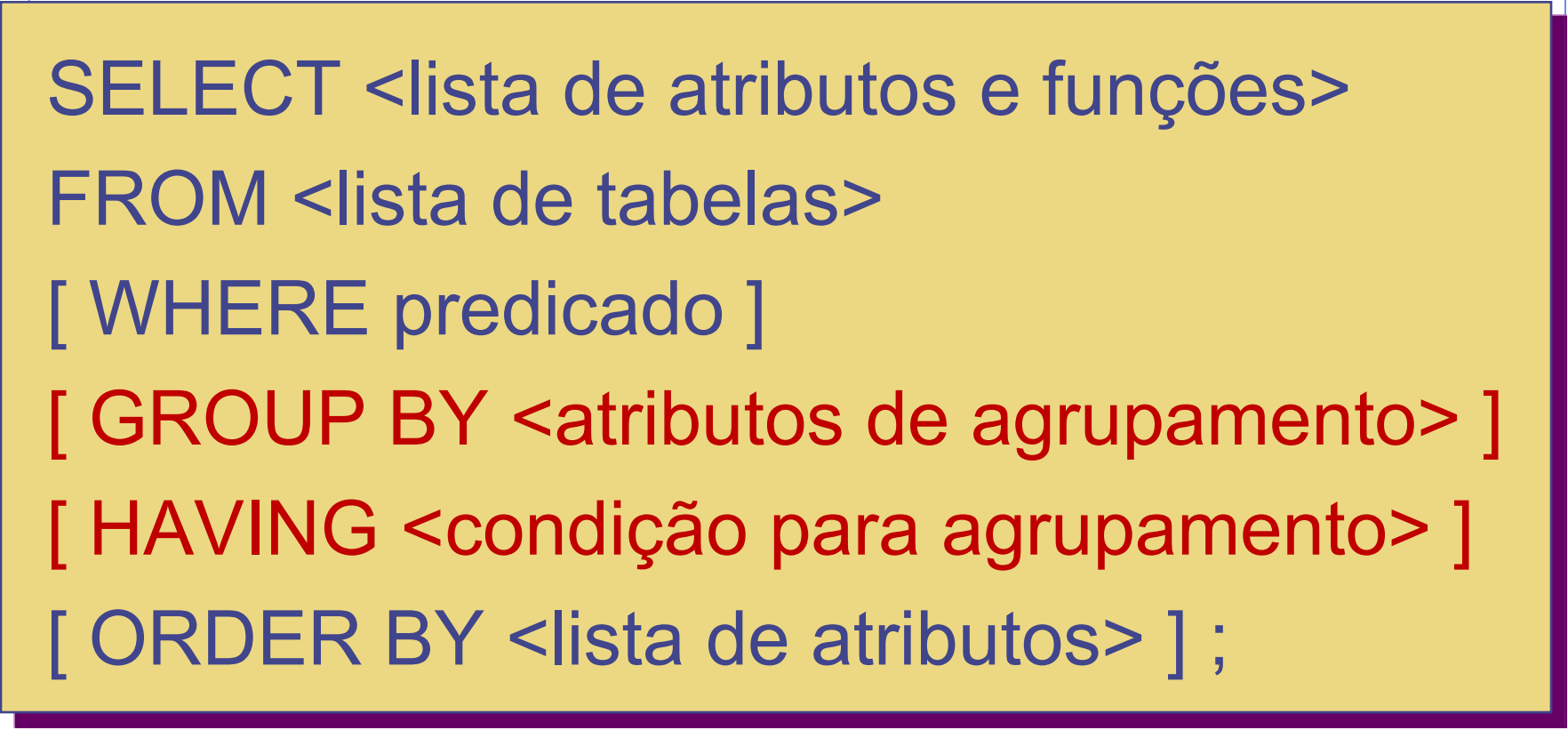
```
SELECT COUNT(DISTINCT COD_AUTOR_ESC) AS AUTORES  
FROM ESCREVE;
```

Output pane

	cod_livro_esc smallint	cod_autor_esc smallint
1	31	1
2	55	1
3	63	1
4	14	2
5	13	3

	autores bigint
1	3

SELECT



```
SELECT <lista de atributos e funções>  
FROM <lista de tabelas>  
[ WHERE predicado ]  
[ GROUP BY <atributos de agrupamento> ]  
[ HAVING <condição para agrupamento> ]  
[ ORDER BY <lista de atributos> ] ;
```

<http://www.postgresql.org/docs/9.2/static/sql-select.html>

Cláusula GROUP BY

- ◆ Podemos dividir o conjunto de tuplas de uma relação em grupos de acordo com algum critério, baseado nos valores dos atributos
 - Por exemplo, na tabela abaixo as tuplas podem ser agrupadas de acordo com o nome do autor
 - Exemplo: PAULO COELHO, MACHADO DE ASSIS e JOSÉ MARIA

Output pane		
Data Output Explain Messages History		
	titulo character(35)	nome character varying(20)
1	MAKTUB	PAULO COELHO
2	BRIDA	PAULO COELHO
3	HISTÓRIAS PARA PAIS, FILHOS E NETOS	PAULO COELHO
4	DOM CASMURRO	MACHADO DE ASSIS
5	SQL	JOSÉ MARIA

Cláusula GROUP BY

```
-- autores de todos os livros da livraria
```

```
SELECT NOME
FROM LIVRO, AUTOR, ESCRIVE
WHERE COD_AUTOR = COD_AUTOR_ESC AND
      COD_LIVRO_ESC = COD_LIVRO
GROUP BY NOME;
```

Data Output		Explain	Messa
	nome		
	character varying(20)		
1	PAULO COELHO		
2	MACHADO DE ASSIS		
3	JOSÉ MARIA		

- Observe que na cláusula **SELECT** só podem constar os atributos presentes no **GROUP BY**
- Isso faz sentido pois, por exemplo, existem 3 livros do PAULO COELHO cadastros, qual deles apareceria no resultado?
- Tente adicionar o campo TITULO na cláusula **SELECT** do comando acima e observe o resultado

Cláusula GROUP BY

-- autores de todos os livros da livraria

```
SELECT NOME
  FROM LIVRO, AUTOR, ESCREVE
 WHERE COD_AUTOR = COD_AUTOR_ESC AND
        COD_LIVRO_ESC = COD_LIVRO
GROUP BY NOME;
```

Data Output	Explain	Messa
	nome character varying(20)	
1	PAULO COELHO	
2	MACHADO DE ASSIS	
3	JOSÉ MARIA	

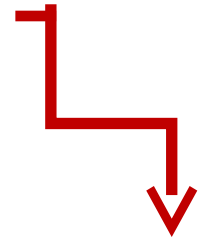
- Tente adicionar o campo TITULO na cláusula **SELECT** do comando acima e observe o resultado
- ERRO: coluna "livro.titulo" deve aparecer na cláusula GROUP BY ou ser utilizada em uma função de agregação

Cláusula GROUP BY

	titulo character(35)	nome character varying(20)	valor numeric(7,2)
1	SQL	JOSÉ MARIA	10.90
2	DOM CASMURRO	MACHADO DE ASSIS	10.90
3	O ALIENISTA	MACHADO DE ASSIS	10.90
4	CONTOS FLUMINENSES	MACHADO DE ASSIS	24.00
5	CONTOS	MACHADO DE ASSIS	24.00
6	MAKTUB	PAULO COELHO	24.00
7	BRIDA	PAULO COELHO	29.50
8	HISTÓRIAS PARA PAIS, FILHOS E NETOS	PAULO COELHO	35.00

Cláusula GROUP BY

	titulo character(35)	nome character varying(20)	valor numeric(7,2)
1	SQL	JOSÉ MARIA	10.90
2	DOM CASMURRO	MACHADO DE ASSIS	10.90
3	O ALIENISTA	MACHADO DE ASSIS	10.90
4	CONTOS FLUMINENSES	MACHADO DE ASSIS	24.00
5	CONTOS	MACHADO DE ASSIS	24.00
6	MAKTUB	PAULO COELHO	24.00
7	BRIDA	PAULO COELHO	29.50
8	HISTÓRIAS PARA PAIS, FILHOS E NETOS	PAULO COELHO	35.00



	nome character varying(20)	valor numeric(7,2)
1	JOSÉ MARIA	10.90
2	MACHADO DE ASSIS	10.90
3	MACHADO DE ASSIS	24.00
4	PAULO COELHO	24.00
5	PAULO COELHO	29.50
6	PAULO COELHO	35.00

Cláusula GROUP BY

- ◆ As funções agregadas (e.g., COUNT, MIN, MAX, AVG) podem ser usadas para cálculos com subgrupos de tuplas definidos pela cláusula GROUP BY

-- qual o número de livros por autor?

```
SELECT NOME, COUNT(*) FROM AUTOR, ESCRIVE
WHERE COD_AUTOR = COD_AUTOR_ESC
GROUP BY NOME
```

	titulo character(35)	nome character varying(20)	valor numeric(7,2)
1	SQL	JOSÉ MARIA	10.90
2	DOM CASMURRO	MACHADO DE ASSIS	10.90
3	O ALIENISTA	MACHADO DE ASSIS	10.90
4	CONTOS FLUMINENSES	MACHADO DE ASSIS	24.00
5	CONTOS	MACHADO DE ASSIS	24.00
6	MAKTUB	PAULO COELHO	24.00
7	BRIDA	PAULO COELHO	29.50
8	HISTÓRIAS PARA PAIS, FILHOS E NETOS	PAULO COELHO	35.00

Cláusula GROUP BY

- ◆ As funções agregadas (e.g., COUNT, MIN, MAX, AVG) podem ser usadas para cálculos com subgrupos de tuplas definidos pela cláusula GROUP BY

-- qual o número de livros por autor?

```
SELECT NOME, COUNT(*) FROM AUTOR, ESCRIVE  
WHERE COD_AUTOR = COD_AUTOR_ESC  
GROUP BY NOME
```

Data Output	Explain	Messages	History
	titulo character(35)	nome character va	
1	SQL	JOSÉ MARIA	10.90
2	DOM CASMURRO	MACHADO DE ASSIS	10.90
3	O ALIENISTA	MACHADO DE ASSIS	24.00
4	CONTOS FLUMINENSES	MACHADO DE ASSIS	24.00
5	CONTOS	MACHADO DE ASSIS	24.00
6	MAKTUB	PAULO COELHO	29.50
7	BRIDA	PAULO COELHO	35.00
8	HISTÓRIAS PARA PAIS, FILHOS E NETOS	PAULO COELHO	

Data Output	Explain	Messages
	nome character varying(20)	count bigint
1	PAULO COELHO	3
2	MACHADO DE ASSIS	4
3	JOSÉ MARIA	1

Cláusula HAVING

- ◆ **HAVING:** é semelhante à cláusula WHERE. HAVING elimina tuplas *agrupadas que não satisfazem a uma determinada condição*
- ◆ **Diferença com WHERE:** WHERE filtra tuplas individuais antes da aplicação do GROUP BY, enquanto HAVING filtra grupo de tuplas criadas por GROUP BY. As condições de filtragem do HAVING devem ser feitas baseando-se nos atributos agrupados por GROUP BY

Cláusula HAVING

-- Editoras cujo total de publicações é maior que 1

```
SELECT COD_EDITORA AS EDITORA
FROM LIVRO
GROUP BY COD_EDITORA
HAVING COUNT(COD_EDITORA) > 1;
```

-- ERRO comum

```
SELECT COD_EDITORA AS EDITORA
FROM LIVRO
GROUP BY COD_EDITORA
HAVING COUNT(EDITORA) > 1;
```

-- ERRO: coluna "editora" não existe

LINE 4: HAVING COUNT(EDITORA) > 1;

-- Selecionar o nome das editoras cujo total de publicações é maior que 1 ?

Cláusula HAVING

-- Selecionar o nome das editoras cujo total de publicações é maior que 1 ?

-- Solução

```
SELECT RAZAO, L.COD_EDITORA
  FROM EDITORA E, LIVRO L
 WHERE E.COD_EDITORA = L.COD_EDITORA
GROUP BY RAZAO, L.COD_EDITORA
HAVING COUNT(L.COD_EDITORA) > 1;
```

Subconsultas: SELECTs aninhados

- ◆ São blocos SELECT...FROM...WHERE completos dentro da cláusula WHERE de outra consulta
- ◆ Essa construção possibilita a realização de consultas sobre os resultados obtidos em outras consultas
- ◆ Podem aparecer na cláusulas
 - SELECT
 - FROM
 - **WHERE**

Subconsultas: SELECTs aninhados

◆ Na cláusula WHERE: subconsulta aninhada

- **não correlacionada**; ou
- correlacionada: avaliada 1 vez para cada linha processada pela consulta superior

◆ Exemplo:

/ Selecionar o título com valor mais alto */*

SELECT TITULO, VALOR

FROM LIVRO

WHERE VALOR **IN** (**SELECT** **MAX**(VALOR)

FROM LIVRO);

Subconsultas: SELECTs aninhados

◆ Exemplo – subconsulta não correlacionada:

/ Selecionar o título com valor mais alto */*

The screenshot shows the PostgreSQL SQL Editor interface. The query entered is:

```
set search_path to livraria;  
  
SELECT TITULO, VALOR  
FROM LIVRO  
WHERE VALOR IN (SELECT MAX (VALOR)  
                FROM LIVRO );
```

The output pane shows the results of the query:

	max numeric
1	35.00

The screenshot shows the same PostgreSQL SQL Editor interface with the same query. The output pane shows the results in a more detailed format:

	titulo character(35)	valor numeric(7,2)
1	HISTÓRIAS PARA PAIS,	35.00

Subconsultas: SELECTs aninhados

◆ Outros exemplos:

/ Selecionar os títulos mais baratos do que o título cujo código é 31 */*

```
SELECT TITULO, VALOR
FROM LIVRO
WHERE VALOR < (SELECT VALOR
                FROM LIVRO
                WHERE COD_LIVRO = 31) ;
```

Subconsultas: SELECTs aninhados

◆ Outros exemplos:

/ Selecionar os títulos mais baratos do que o título cujo código é 31 */*

```
SELECT TITULO, VALOR
FROM LIVRO
WHERE VALOR < (SELECT VALOR
FROM LIVRO
WHERE COD_LIVRO = 31) ;
```

Output pane

Data Output	Explain	Messages	History
	valor numeric(7,2)		
1	24.00		

Subconsultas: SELECTs aninhados

◆ Outros exemplos:

/ Selecionar os títulos mais baratos do que o título cujo código é 31 */*

```
SELECT TITULO, VALOR
FROM LIVRO
WHERE VALOR < (SELECT VALOR
                FROM LIVRO
                WHERE COD_LIVRO = 31) ;
```

Output pane			
Data Output		Explain	Messages
		History	
	titulo character(35)	valor numeric(7,2)	
1	DOM CASMURRO	10.90	
2	SQL	10.90	

Subconsultas: SELECTs aninhados

◆ Outros exemplos:

/ Selecionar os títulos de livros da editora com
COD_EDITORA = 3 que possuam valor menor do que todos
os livros da editora com COD_EDITORA = 1*/*

```
SELECT TITULO, VALOR
FROM LIVRO
WHERE COD_EDITORA = 3 AND
      VALOR < ALL (SELECT VALOR
                   FROM LIVRO
                   WHERE COD_EDITORA = 1);
```

Subconsultas: SELECTs aninhados

◆ Outros exemplos:

/ Selecionar os títulos de livros da editora com
COD_EDITORA = 3 que possuam valor menor do que todos
os livros da editora com COD_EDITORA = 1*/*

```
SELECT TITULO, VALOR
FROM LIVRO
WHERE COD_EDITORA = 3 AND
      VALOR < ALL(SELECT VALOR
                    FROM LIVRO
                    WHERE COD_EDITORA = 1);
```

Output pane

	valor numeric(7,2)
1	24.00
2	29.50

Output pane

	titulo character(35)	valor numeric(7,2)
1	DOM CASMURRO	10.90

Subconsultas: SELECTs aninhados

◆ Pode-se utilizar com os operadores ANY e ALL

- >
- >=
- <
- <=
- =
- <>

◆ Curiosidade: = ANY equivale ao IN

SQL DML

◆ SELECT ... FROM ... WHERE ...

- lista atributos de uma ou mais tabelas de acordo com alguma condição
- Detalhamento sobre JUNÇÕES

◆ INSERT INTO ...

- insere dados em uma tabela

◆ DELETE FROM ... WHERE ...

- remove dados de tabelas já existentes

◆ UPDATE ... SET ... WHERE ...

- altera dados específicos de uma tabela

Junção Natural

◆ SQL (primeiras versões)

- não tem uma representação para a operação de junção

◆ Definida em termos de

- um produto cartesiano
- uma seleção
- uma projeção

Junção

- ◆ Não é representada explicitamente
- ◆ Cláusulas SELECT e WHERE
 - especificam atributos com mesmo nome usando o nome da tabela e o nome do atributo (nome_tabela.nome_atributo)
- ◆ Cláusula FROM
 - possui mais do que uma tabela
- ◆ Cláusula WHERE
 - inclui as condições de junção

Junção

Relembrando os tipos

◆ Existem **dois tipos de condição de junção**

- **Equi-junções** empregam o operador de igualdade
- **Junções theta** empregam outros operadores como $<$, $>$, BETWEEN, etc.

◆ Existem **três tipos de junção**

- **Junção interna:** retornam uma linha somente quando os atributos na junção contêm valores que satisfazem a condição de junção
- **Junção externa:** retornam uma linha mesmo quando um dos atributos (ou ambas) na condição de junção contém um valor nulo
- **Auto junção:** retornam o resultado da junção de uma tabela com ela mesma

Nota: exemplos de junção interna

Exemplos

-- Selecionar todos os títulos dos livros da editora ROCCO

```
SELECT LI.TITULO
FROM EDITORA ED, LIVRO LI
WHERE (ED.RAZAO = 'ROCCO') AND
      (ED.COD_EDITORA = LI.COD_EDITORA);
```

Output pane

	cod_editora smallint	razao character varying(20)	endereco character(20)	cidade character(70)
1	1	ROCCO	R. RODRIGO S	RIO DE JANEI
2	2	GLOBO		RIO DE JANEI
3	3	ATICA		SÃO PAULO
4	4	USP/ICMC		

SELECT * FROM EDITORA

SELECT * FROM LIVRO

	titulo character(35)	valor numeric(7,2)	volume smallint	cod_editora smallint
1	MAKTUB	24.00		1
2	BRIDA	29.50		1
3	HISTÓRIAS PA	35.00		2
4	DOM CASMURRO	10.90		3
5	SQL	10.90		4

Exemplos

/ Selecionar os títulos da tabela livro e os nomes da tabela autor */*

```
SELECT AU.NOME, LI.TITULO
FROM AUTOR AU, LIVRO LI, ESCRIVE ES
WHERE (AU.NOME = 'PAULO COELHO') AND
      (AU.COD_AUTOR = ES.COD_AUTOR_ESC) AND
      (LI.COD_LIVRO = ES.COD_LIVRO_ESC);
```

	cod_autor smallint	nome character varying(20)	nascimento date
1	1	PAULO COELHO	1947-01-01
2	2	MACHADO DE ASSIS	1839-06-21
3	3	JOSÉ MARIA	1960-08-25

AUTOR

LIVRO

	cod_livro smallint	titulo character(35)	valor numeric(7,2)	volume smallint	cod_editora smallint
1	31	MAKTUB	24.00		1
2	55	BRIDA	29.50		1
3	63	HISTÓRIAS P	35.00		2
4	14	DOM CASMURR	10.90		3
5	13	BQL	10.90		4

	cod_livro_esc smallint	cod_autor_esc smallint
1	31	1
2	55	1
3	63	1
4	14	2
5	13	3

ESCREVE

Auto Junção

- ◆ É necessário empregar um apelido (*alias*) para identificar cada referencia para a tabela na consulta

Cláusula AS

◆ Renomeia

- atributos
 - ◆ deve aparecer na cláusula SELECT
 - ◆ **útil para a visualização das respostas na tela**
- relações
 - ◆ deve aparecer na cláusula FROM
 - ◆ **útil quando a mesma relação é utilizada mais do que uma vez na mesma consulta**

◆ Sintaxe

- nome_antigo AS nome_novo

Auto Junção


Nota: exemplo de auto junção

Auxiliada pelo uso de apelidos

◆ Exemplo:

- Para selecionar o nome de todos os funcionários de uma empresa juntamente com o nome de seus gerentes:

- EMPREGADO = (id_funcionario, nome, data_nascimento, endereço, cpf, id_gerente)



```
SELECT t1.nome funcionario,  
       t2.nome gerente  
FROM   empregado t1, empregado t2  
WHERE  t1.id_gerente = t2.id_funcionario;
```

Auto Junção

Nota: exemplo de auto junção

Auxiliada pelo uso de apelidos

◆ Exemplo auto junção

Empregado

id	nome	datanasc	end	cpf	gerente
1	José	05/01/1977	R. X, 111	111.111.111-11	NULL
2	Maria	07/07/1978	R. Y, 22	222.222.222-22	1
3	João	11/01/1984	R. Z, 78	333.333.333-33	2
4	Pedro	03/08/1954	R. W, 10	444.444.444-44	1

Auto Junção

Nota: exemplo de auto junção

Auxiliada pelo uso de apelidos

Exemplo auto junção

Junção de Empregado **t1** e Empregado **t2**

t1.id	t1.nome	t1.datanasc	...	t1.gerente	t2.id	t2.nome	t2.datanasc	...	t2.gerente
2	Maria	07/07/1978	...	1	1	José	05/01/1977	...	NULL
3	João	11/01/1984	...	2	2	Maria	07/07/1978	...	1
4	Pedro	03/08/1954	...	1	1	José	05/01/1977	...	NULL

Junção na cláusula FROM

◆ SQL-92

- inclusão de operações adicionais na cláusula **FROM**

◆ Operações adicionais no PostgreSQL

- ... [INNER] JOIN ... ON ...
- ... LEFT [OUTER] JOIN ... ON ...
- ... RIGHT [OUTER] JOIN ... ON ...
- ... FULL [OUTER] JOIN ... ON ...
- CROSS JOIN

Nota: exemplos de junção interna

Exemplos

-- Selecionar todos os títulos dos livros da editora ROCCO

```
SELECT LI.TITULO
FROM EDITORA ED INNER JOIN LIVRO LI ON
      ED.COD_EDITORA = LI.COD_EDITORA
WHERE (ED.RAZAO = 'ROCCO');
```

Output pane

	cod_editora smallint	razao character varying(20)	endereco character(20)	cidade character(70)
1	1	ROCCO	R. RODRIGO S	RIO DE JANEI
2	2	GLOBO		RIO DE JANEI
3	3	ATICA		SÃO PAULO
4	4	USP/ICMC		

SELECT * FROM EDITORA

SELECT * FROM LIVRO

		titulo character(35)	valor numeric(7,2)	volume smallint	cod_editora smallint
1	31	MAKTUB	24.00		1
2	55	BRIDA	29.50		1
3	63	HISTÓRIAS PA	35.00		2
4	14	DOM CASMURRO	10.90		3
5	13	SQL	10.90		4

Exemplos

/ Selecionar os títulos da tabela livro e os nomes da tabela autor */*

?

	cod_autor smallint	nome character varying(20)	nascimento date
1	1	PAULO COELHO	1947-01-01
2	2	MACHADO DE ASSIS	1839-06-21
3	3	JOSÉ MARIA	1960-08-25

AUTOR

	cod_livro smallint	titulo character(35)	valor numeric(7,2)	volume smallint	cod_editora smallint
1	31	MAKTUB	24.00		1
2	55	BRIDA	29.50		1
3	63	HISTÓRIAS P	35.00		2
4	14	DOM CASMURR	10.90		3
5	13	SQL	10.90		4

LIVRO

	cod_livro_esc smallint	cod_autor_esc smallint
1	31	1
2	55	1
3	63	1
4	14	2
5	13	3

ESCREVE

Exemplos

/ Selecionar os títulos da tabela livro e os nomes da tabela autor */*

```
SELECT AU.NOME, LI.TITULO
FROM AUTOR AU INNER JOIN
      (LIVRO LI INNER JOIN ESCREVE ES
       ON LI.COD_LIVRO = ES.COD_LIVRO_ESC)
ON AU.COD_AUTOR = ES.COD_AUTOR_ESC
WHERE (AU.NOME = 'PAULO COELHO');
```

Data Output	Explain	Messages	History
	cod_autor smallint	nome character varying(20)	nascimento date
1	1	PAULO COELHO	1947-01-01
2	2	MACHADO DE ASSIS	1839-06-21
3	3	JOSÉ MARIA	1960-08-25

AUTOR

LIVRO

Output pane

Data Output	Explain	Messages	History		
	cod_livro smallint	titulo character(35)	valor numeric(7,2)	volume smallint	cod_editora smallint
1	31	MAKTUB	24.00		1
2	55	BRIDA	29.50		1
3	63	HISTÓRIAS PA	35.00		2
4	14	DOM CASMURRO	10.90		3
5	13	BQL	10.90		4

Data Output	Explain	Messages
	cod_livro_esc smallint	cod_autor_esc smallint
1	31	1
2	55	1
3	63	1
4	14	2
5	13	3

ESCREVE

Junção externa na cláusula FROM

◆ Variante da operação de JOIN que baseia-se em valores NULL. O resultado de um OUTER JOIN é igual a de um INNER JOIN mas com a inclusão das tuplas que não satisfazem a condição de JOIN.

◆ Três variantes:

- LEFT OUTER JOIN

- ◆ As tuplas da tabela à esquerda que não obedecem a condição do JOIN aparecem na resposta

- RIGHT OUTER JOIN

- ◆ As tuplas da tabela à direita que não obedecem a condição do JOIN aparecem na resposta

- FULL OUTER JOIN

- ◆ As tuplas das duas tabelas que não obedecem a condição do JOIN aparecem na resposta

Junção externa na cláusula FROM



```
/* Criando as tabelas*/
```

```
CREATE TABLE orientador (  
    id INT PRIMARY KEY,  
    nome VARCHAR(255)  
);
```

```
CREATE TABLE aluno (  
    matricula INT PRIMARY KEY,  
    nome VARCHAR(255),  
    orientador_id INT REFERENCES orientador(id)  
);
```

Junção externa na cláusula FROM



-- Povoando as tabelas

```
INSERT INTO orientador VALUES (1, 'Prof. Luiz'),  
                                (2, 'Profa. Maria');
```

```
INSERT INTO aluno VALUES (1, 'Alessandra', NULL),  
                           (2, 'Walter', 2),  
                           (3, 'Jéssica', NULL);
```

Junção externa na cláusula FROM



/* execute as seguintes junções nessas duas tabelas e observe os resultados */

- INNER JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN
- CROSS JOIN



Data Output	Explain	Messages	History
	id integer	nome character varying(255)	
1	1	Prof. Luiz	
2	2	Profa. Maria	

Data Output	Explain	Messages	History
	matricula integer	nome character varying(255)	orientador_id integer
1	1	Alessandra	
2	2	Walter	2
3	3	Jéssica	

```

SELECT *
FROM orientador INNER JOIN aluno
ON aluno.orientador_id = orientador.id;
  
```

Data Output

Explain

Messages

History

	id integer	nome character varying(255)	matricula integer	nome character varying(255)	orientador_id integer
1	2	Profa. Maria	2	Walter	2



	Data Output	Explain	Messages	History
	id integer	nome character varying(255)		
1	1	Prof. Luiz		
2	2	Profa. Maria		

	Data Output	Explain	Messages	History
	matricula integer	nome character varying(255)	orientador_id integer	
1	1	Alessandra		
2	2	Walter	2	
3	3	Jéssica		

```

SELECT *
FROM orientador LEFT OUTER JOIN aluno
ON aluno.orientador_id = orientador.id;
  
```

Data Output

Explain

Messages

History

	id integer	nome character varying(255)	matricula integer	nome character varying(255)	orientador_id integer
1	1	Prof. Luiz			
2	2	Profa. Maria	2	Walter	2



Data Output	Explain	Messages	History
	id integer	nome character varying(255)	
1	1	Prof. Luiz	
2	2	Profa. Maria	

Data Output	Explain	Messages	History
	matricula integer	nome character varying(255)	orientador_id integer
1	1	Alessandra	
2	2	Walter	2
3	3	Jéssica	

```

SELECT *
FROM orientador RIGHT OUTER JOIN aluno
ON aluno.orientador_id = orientador.id;
  
```

	id integer	nome character varying(255)	matricula integer	nome character varying(255)	orientador_id integer
1			1	Alessandra	
2	2	Profa. Maria	2	Walter	2
3			3	Jéssica	



Data Output	Explain	Messages	History
	id integer	nome character varying(255)	
1	1	Prof. Luiz	
2	2	Profa. Maria	

Data Output	Explain	Messages	History
	matricula integer	nome character varying(255)	orientador_id integer
1	1	Alessandra	
2	2	Walter	2
3	3	Jéssica	

```

SELECT *
FROM orientador FULL OUTER JOIN aluno
ON aluno.orientador_id = orientador.id;
  
```

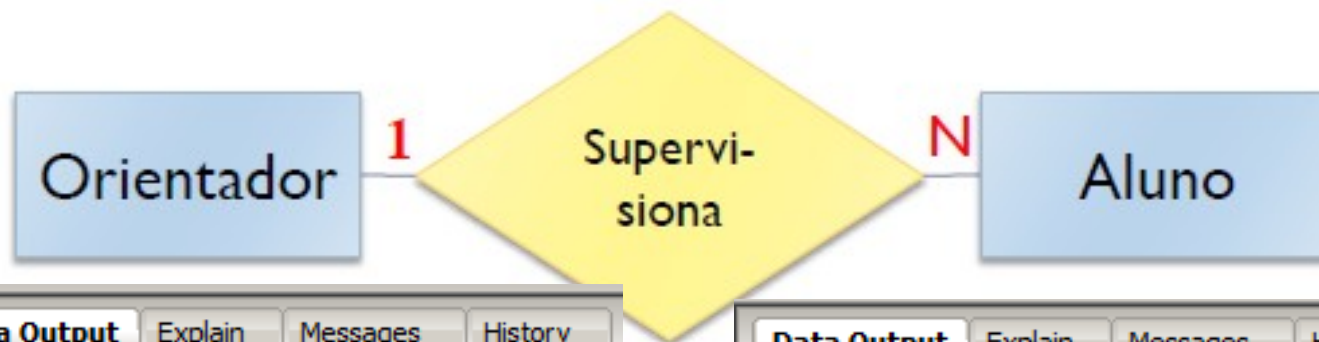
Data Output

Explain

Messages

History

	id integer	nome character varying(255)	matricula integer	nome character varying(255)	orientador_id integer
1	1	Prof. Luiz			
2	2	Profa. Maria	2	Walter	2
3			3	Jéssica	
4			1	Alessandra	



Data Output			
	id integer	nome character varying(255)	
1	1	Prof. Luiz	
2	2	Profa. Maria	

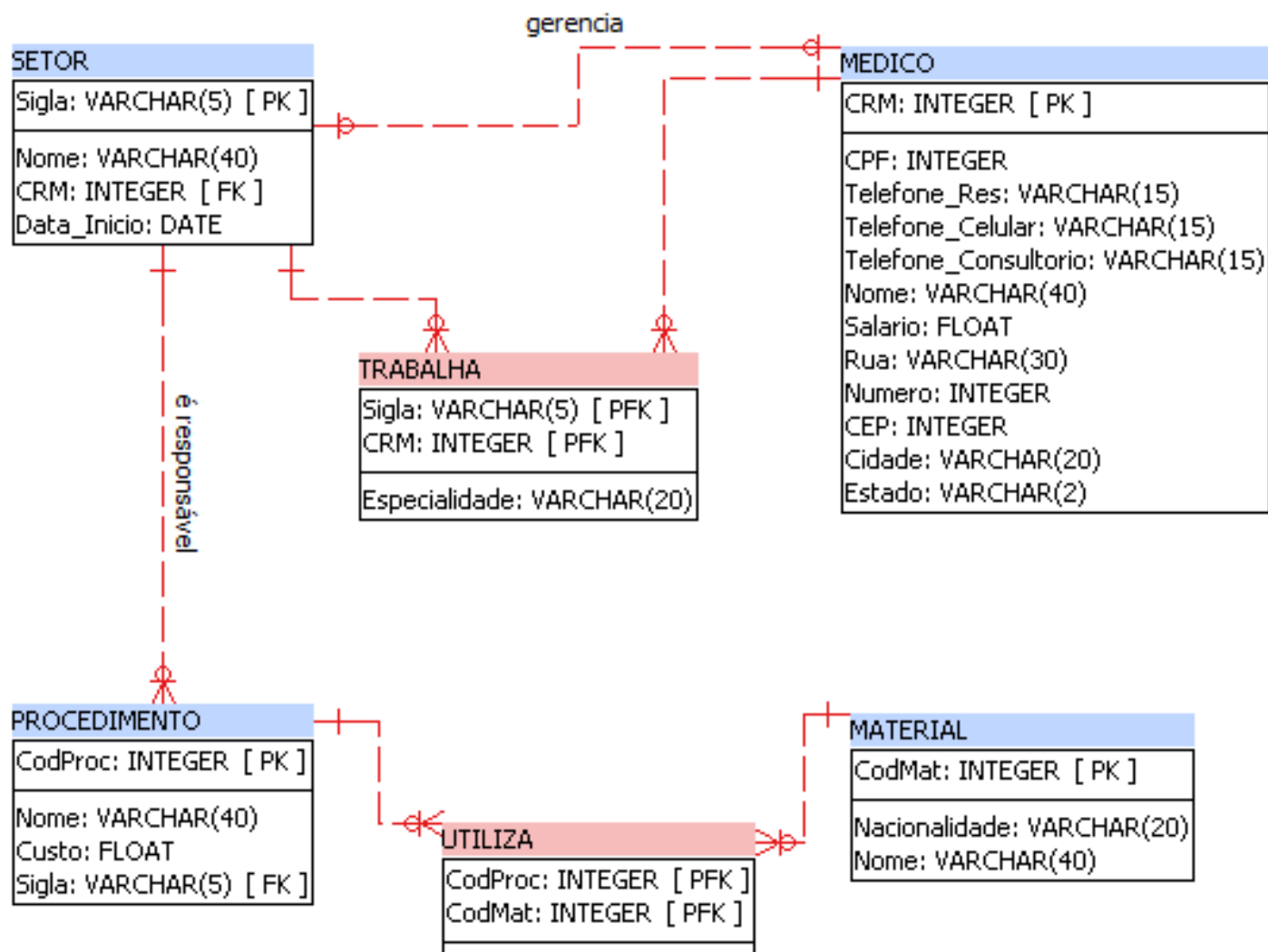
Data Output			
	matricula integer	nome character varying(255)	orientador_id integer
1	1	Alessandra	
2	2	Walter	2
3	3	Jéssica	

```

SELECT *
FROM orientador CROSS JOIN aluno;
  
```

Data Output					
	id integer	nome character varying(255)	matricula integer	nome character varying(255)	orientador_id integer
1	1	Prof. Luiz	1	Alessandra	
2	1	Prof. Luiz	2	Walter	2
3	1	Prof. Luiz	3	Jéssica	
4	2	Profa. Maria	1	Alessandra	
5	2	Profa. Maria	2	Walter	2
6	2	Profa. Maria	3	Jéssica	

Exercício



Exercício

◆ Escreva consultas em SQL

1. Listar a quantidade total de médicos
2. Listar o nome dos procedimentos e quantos materiais cada um utiliza
3. Listar a média de salário por setor
4. Listar o nome e o custo do procedimento mais barato
5. Listar os nomes dos setores nos quais existam mais de 10 médicos trabalhando

Bibliografia

- ◆ Elmasri, Ramez; Navathe, Shamkant B. **Sistemas de banco de dados**. 4 ed. São Paulo: Addison Wesley, 2005, 724 p. Bibliografia: p. [690]-714.

Leitura complementar para casa

- ◆ Capítulo 4 do livro: Elmasri, Ramez; Navathe, Shamkant B. Sistemas de banco de dados. 6ª edição
- ◆ Manual do SGBD PostgreSQL
 - <http://www.postgresql.org/docs/manuals/>
 - Explorar as outras particularidades dos comandos apresentados

Conteúdo Segunda Prova

◆ Álgebra Relacional

◆ SQL (tudo sobre SQL)

- DDL
- DML

◆ **Data da Prova: 31/05**