
Banco de Dados – IMD0401

Aula 15 – SQL

Data Query Language

João Carlos Xavier Júnior

jcxavier@imd.ufrn.br

SQL - DQL

❑ Seleção:

- ❖ A forma básica do comando de seleção é:

```
SELECT <lista de atributos>  
FROM   <lista de tabelas>  
WHERE  <condição>
```

- ❖ Onde:

- **<lista de atributos>** é uma lista de nomes de atributos cujos valores serão ser recuperados pela consulta.
- **<lista de tabelas>** é uma lista de nomes de relações requeridas para processar a consulta.
- **<condição>** é uma expressão (lógica) que identifica as tuplas a serem recuperadas pela consulta.

SQL - DQL

❑ Seleção:

❖ Exemplo:

```
select nome, cpf, genero from Cliente
```

❖ Resultado:

Output pane			
Data Output Explain Messages History			
	nome character varying(45)	cpf character varying(14)	genero character(1)
1	Roberto Carlos	111.111.111-01	M
2	Ana Maria	222.222.222-02	F
3	Francisco dos Santos	333.333.333-03	M
4	Angelina Jolie	444.444.444-04	F

SQL - DQL

❑ Seleção:

❖ **Observação:** sempre use **filtros** em sua seleção.

```
select Nome, cpf, sexo from Cliente  
where cidade = 'Natal'  
and genero = 'M'
```

❖ Resultado:

Data Output Explain Messages History			
	nome character varying(45)	cpf character varying(14)	genero character(1)
1	Roberto Carlos	111.111.111-01	M


SQL - DQL

❑ Seleção:

❖ **Observação:** nunca faça isso.

select * from Cliente

❖ Resultado:



	idcliente integer	nome character varying(45)	cpf character varying(14)	genero character(1)	cidade character varying(15)
1	1	Roberto Carlos	111.111.111-01	M	Natal
2	2	Ana Maria	222.222.222-02	F	Parnamirim
3	3	Francisco dos Sa	333.333.333-03	M	
4	4	Angelina Jolie	444.444.444-04	F	

SQL - DQL

❑ Seleção:

- ❖ A cláusula **SELECT** permite duplicação de resultados nas consultas.
- ❖ Para forçar a eliminação de duplicação, acrescenta-se a palavra chave **DISTINCT**.

❖ Exemplo:

```
select genero from Cliente  
select distinct genero from Cliente  
select all genero from Cliente
```

SQL - DQL

❑ Seleção (distinct):

❖ Resultado:

```
select genero from Cliente
```

```
select distinct genero from Cliente
```

```
select all genero from Cliente
```

Data Output Explain	
	genero character(1)
1	M
2	F
3	M
4	F

Data Output Explain	
	genero character(1)
1	F
2	M

Data Output Explain	
	genero character(1)
1	M
2	F
3	M
4	F

SQL - DQL

❑ Seleção:

- ❖ A cláusula SELECT pode conter funções que operam sobre uma coleção de valores de uma determinada coluna da tabela.

- ❖ O SQL fornece 5 funções agregadas:
 - COUNT: número de tuplas ou valores.
 - SUM: soma os valores de uma coluna.
 - AVG: calcula a média dos valores de uma coluna.
 - MAX: identifica o maior valor de uma coluna.
 - MIN: identifica o menor valor de uma coluna.

SQL - DQL

❑ Seleção:

❖ Exemplos:

```
select count(genero) from Cliente
```

```
select count(distinct genero) from Cliente
```

Previous queries

```
select count(genero) from Cliente  
select count(distinct genero) from Cliente
```

Output pane

	count bigint
1	4

Previous queries

```
select count(genero) from Cliente  
select count(distinct genero) from Cliente
```

Output pane

	count bigint
1	2

SQL - DQL

❑ Seleção:

❖ A cláusula SELECT permite que colunas possam ser **renomeadas** na consulta. A mesma ideia pode ser usada para tabelas.

❖ Exemplos:

```
select nome, cpf, genero, cidade as  
Localidade from Cliente
```

```
select c.nome, c.cpf, c.genero,  
c.cidade as Localidade from Cliente c
```

SQL - DQL

- ❑ Seleção: renomeando colunas e tabelas.

Previous queries

```
select nome, cpf, genero, cidade as Localidade from Cliente  
select c.nome, c.cpf, c.genero, c.cidade as Localidade from Cliente c
```

<

Output pane

Data Output Explain Messages History

	nome character varying(45)	cpf character varying(14)	genero character(1)	localidade character varying(15)
1	Roberto Carlos	111.111.111-01	M	Natal
2	Ana Maria	222.222.222-02	F	Parnamirim
3	Francisco dos Santos	333.333.333-03	M	
4	Angelina Jolie	444.444.444-04	F	

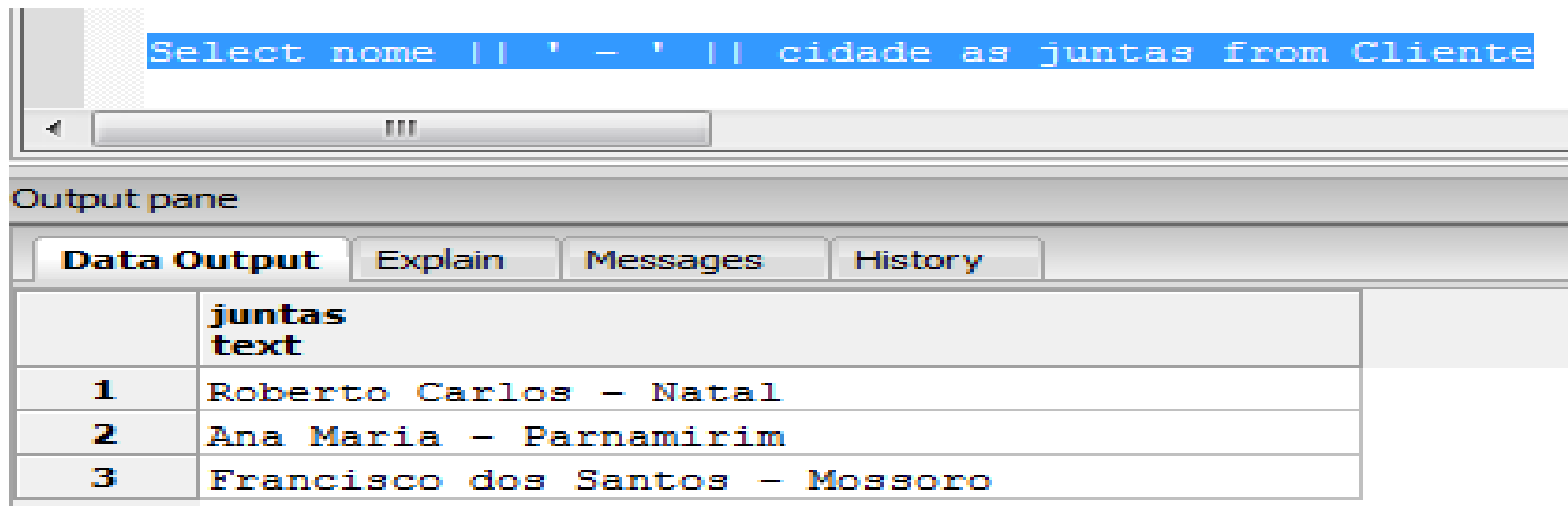
SQL - DQL

❑ Seleção:

❖ SQL também permite função para concatenação (||).

❖ Exemplos:

```
select nome || ' - ' || cidade as  
juntas from Cliente
```



The screenshot shows a SQL query execution window. The query is highlighted in blue: `Select nome || ' - ' || cidade as juntas from Cliente`. Below the query editor is an "Output pane" with tabs for "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is selected, displaying a table with the results of the query.

	juntas text
1	Roberto Carlos - Natal
2	Ana Maria - Parnamirim
3	Francisco dos Santos - Mossoro

SQL - DQL

□ From:

- ❖ A cláusula **FROM** corresponde ao **produto cartesiano** da álgebra relacional.
- ❖ A eliminação das tuplas incoerentes do produto cartesiano é feito através da cláusula **WHERE**.

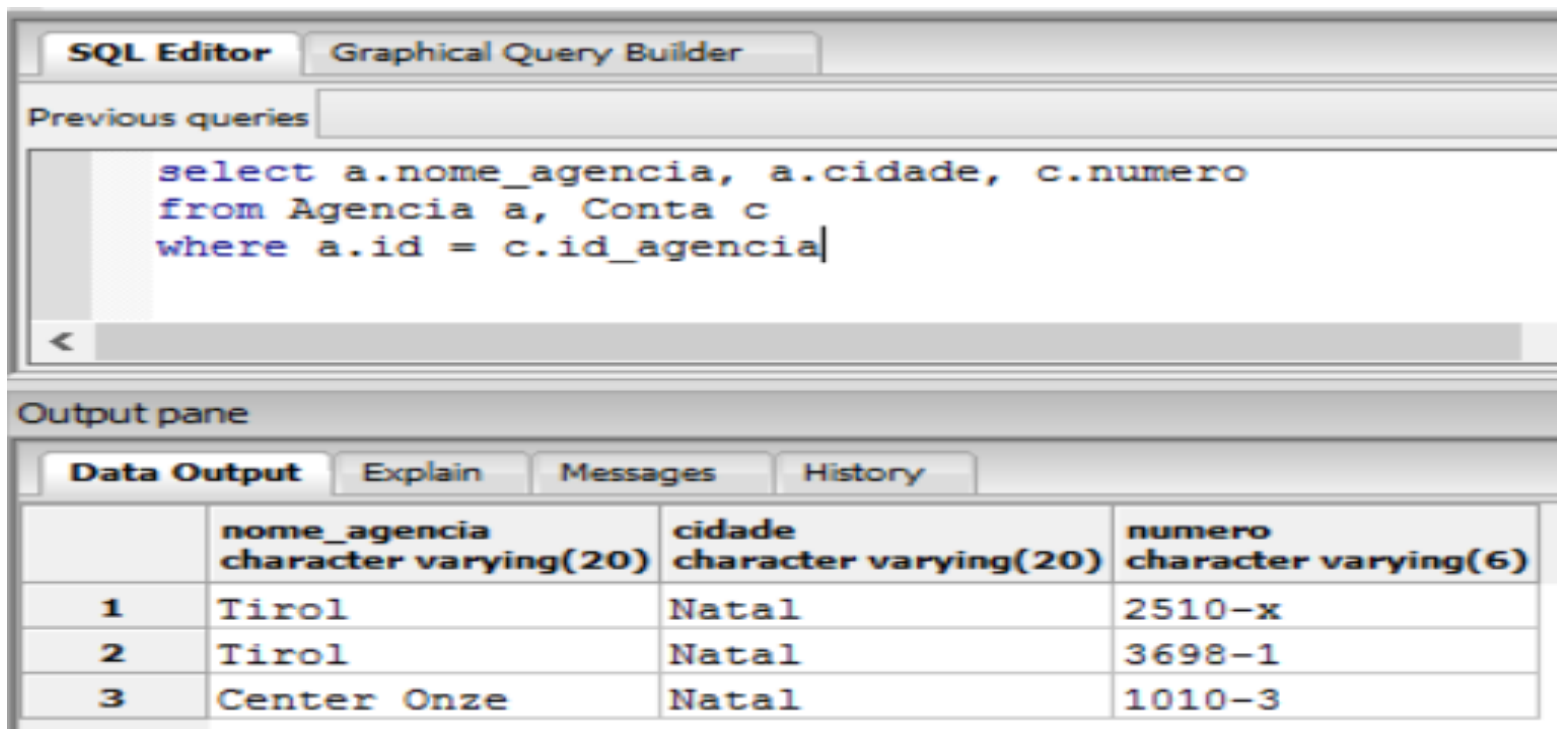
❖ Exemplo:

```
select a.nome_agencia, a.cidade,  
       c.numero  
from Agencia a, Conta c  
where a.id = c.id_agencia
```

SQL - DQL

□ From:

❖ Resultado:



The screenshot shows a database application interface. At the top, there are two tabs: "SQL Editor" and "Graphical Query Builder". Below the tabs is a text area labeled "Previous queries" containing the following SQL query:

```
select a.nome_agencia, a.cidade, c.numero  
from Agencia a, Conta c  
where a.id = c.id_agencia|
```

Below the query editor is an "Output pane" with four tabs: "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is selected, displaying a table with the results of the query.

	nome_agencia character varying(20)	cidade character varying(20)	numero character varying(6)
1	Tirol	Natal	2510-x
2	Tirol	Natal	3698-1
3	Center Onze	Natal	1010-3

SQL - DQL

□ Where:

- ❖ A cláusula **WHERE** usa os conectivos lógicos AND, OR e NOT. É permitido usar expressões aritméticas de comparação (=, <>, <, <=, >=, >).
- ❖ Operador BETWEEN permite que um atributo seja comparado dentro de uma faixa especificada.
- ❖ Exemplo:

```
Select idEmprestimo from Emprestimo  
where total between 90000 and 100000
```

SQL - DQL

□ Where:

❖ Operador **LIKE** permite a comparações em seqüências de caracteres.

❖ Exemplo:

```
select cpf, nome from Cliente  
where nome like '%Mar%'
```

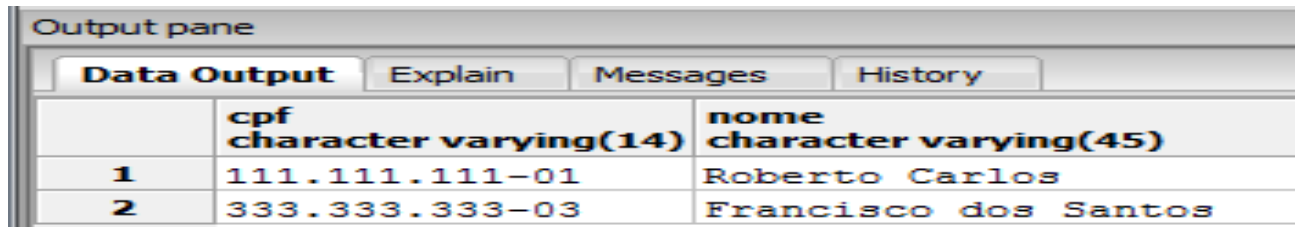
	Data Output	Explain	Messages	History
	cpf character varying(14)	nome character varying(45)		
1	222.222.222-02	Ana Maria		

SQL - DQL

□ Where:

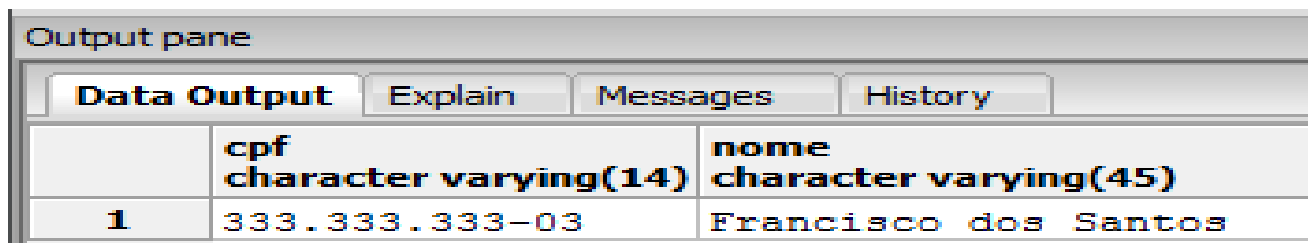
❖ Outros:

```
select cpf, nome from Cliente  
where nome like '%os%'
```



	cpf character varying(14)	nome character varying(45)
1	111.111.111-01	Roberto Carlos
2	333.333.333-03	Francisco dos Santos

```
select cpf, nome from Cliente  
where nome like 'Fr%'
```



	cpf character varying(14)	nome character varying(45)
1	333.333.333-03	Francisco dos Santos

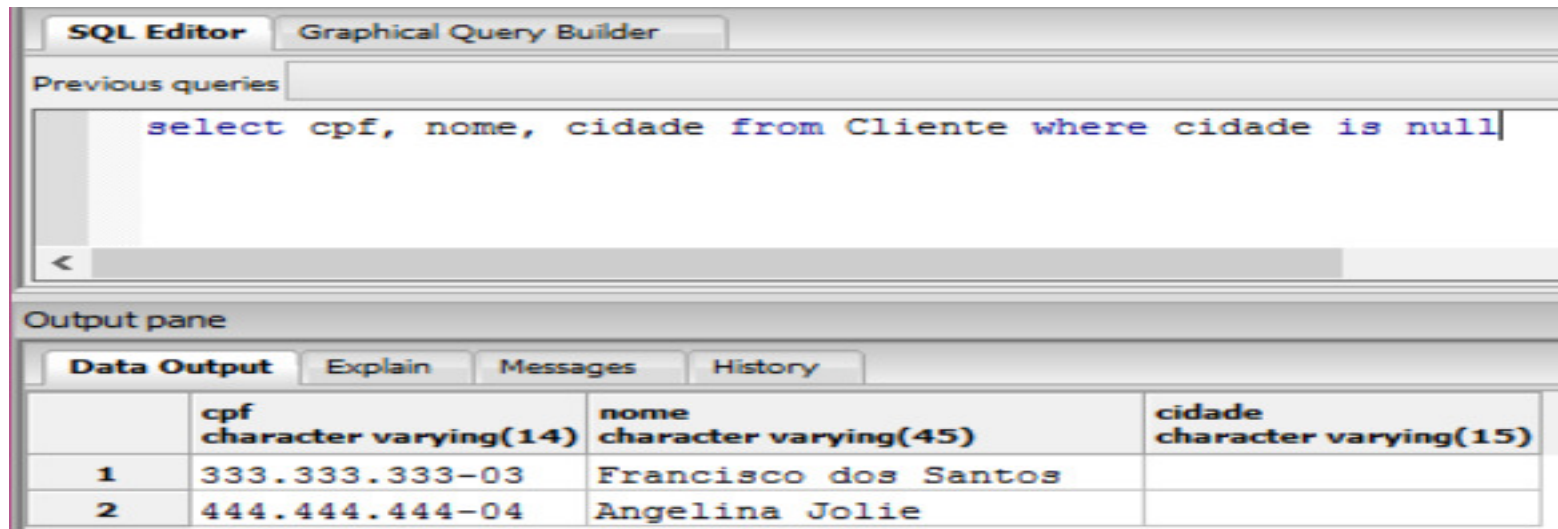
SQL - DQL

□ Where:

❖ Operador **IS NULL** verifica se o atributo é nulo.

❖ Exemplo:

```
select cpf, nome, cidade from Cliente  
where cidade is null
```



The screenshot shows a database application interface. At the top, there are two tabs: "SQL Editor" and "Graphical Query Builder". Below the tabs is a text area containing the SQL query: `select cpf, nome, cidade from Cliente where cidade is null`. Below the text area is a horizontal scrollbar. At the bottom, there is an "Output pane" with four tabs: "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is selected, showing a table with the results of the query. The table has four columns: an index column, "cpf character varying(14)", "nome character varying(45)", and "cidade character varying(15)". There are two rows of data.

	cpf character varying(14)	nome character varying(45)	cidade character varying(15)
1	333.333.333-03	Francisco dos Santos	
2	444.444.444-04	Angelina Jolie	

SQL - DQL

□ Where:

❖ Operador **IS NOT NULL** verifica se o atributo não é nulo.

❖ Exemplo:

```
select cpf, nome, cidade from Cliente  
where cidade is not null
```

Output pane			
Data Output Explain Messages History			
	cpf character varying(14)	nome character varying(45)	cidade character varying(15)
1	111.111.111-01	Roberto Carlos	Natal
2	222.222.222-02	Ana Maria	Parnamirim

SQL - DQL

□ Where:

❖ Operador **IN** permite que um atributo seja comparado com um conjunto.

❖ Exemplo:

```
Select nome, cpf from Cliente  
where uf in ('AC', 'RN')
```

	Data Output	Explain	Messages	History
	nome character varying(45)	cpf character varying(14)		
1	Roberto Carlos	111.111.111-01		
2	Ana Maria	222.222.222-02		
3	Marina Silva	555.555.555-05		

Questões...



SQL-DML e SQL – DQL

❑ Inclusão:

- ❖ Podemos inserir várias tuplas numa relação através de uma consulta (**SELECT**).
- ❖ Exemplo: Exemplo: inserir todas as tuplas de Alunos em ExAlunos.

```
Insert into ExAlunos  
Select Matricula, Nome, Endereco  
From Alunos  
where Matricula > 0
```

SQL-DML e SQL – DQL

□ Exemplo:

pgAdmin III Edit Data - PostgreSQL Database Server 8.2 (localhost:5432) - BD_AULA11 - alunos

File Edit View Help

No limit

	matricula [PK] integer	nome character varying(40)	endereco character varying(40)
1	1	ANA	RUA A
2	2	ISABEL	RUA A
3	3	RAFAEL	RUA C
*			

pgAdmin III Edit Data - PostgreSQL Database Server 8.2 (localhost:5432) - BD_AULA11 - exalunos

File Edit View Help

No limit

	matricula [PK] integer	nome character varying(40)	endereco character varying(40)
*			

SQL-DML e SQL – DQL

❑ Resultado:

pgAdmin III Edit Data - PostgreSQL Database Server 8.2 (localhost:5432) - BD_AULA11 - exalunos

File Edit View Help

Icons: Refresh, Undo, Redo, Print, Copy, Paste, Filter, Help, No limit

	matricula [PK] integer	nome character varying(40)	endereco character varying(40)
1	1	ANA	RUA A
2	2	ISABEL	RUA A
3	3	RAFAEL	RUA C
*			

Questões...



Exercite-se

❑ Esquema relacional:

- ❖ Solicitação (IdSolicitação, IdFuncionario, DataSolicitação).
- ❖ Itens (IdSolicitação, IdProduto, Quantidade).
- ❖ Funcionário (IdFuncionario, Nome).
- ❖ Produtos (IdProduto, Descrição).

❑ Resolva usando SQL-DQL:

- ❖ Mostre a descrição dos produtos que a quantidade solicitada foi maior que 5.000;
- ❖ Mostre o nome dos funcionários, sem repetição, que fizeram solicitações na data de ontem.