

Banco de Dados Aplicado

Cristiano Santos

cristiano.santos@amf.edu.br

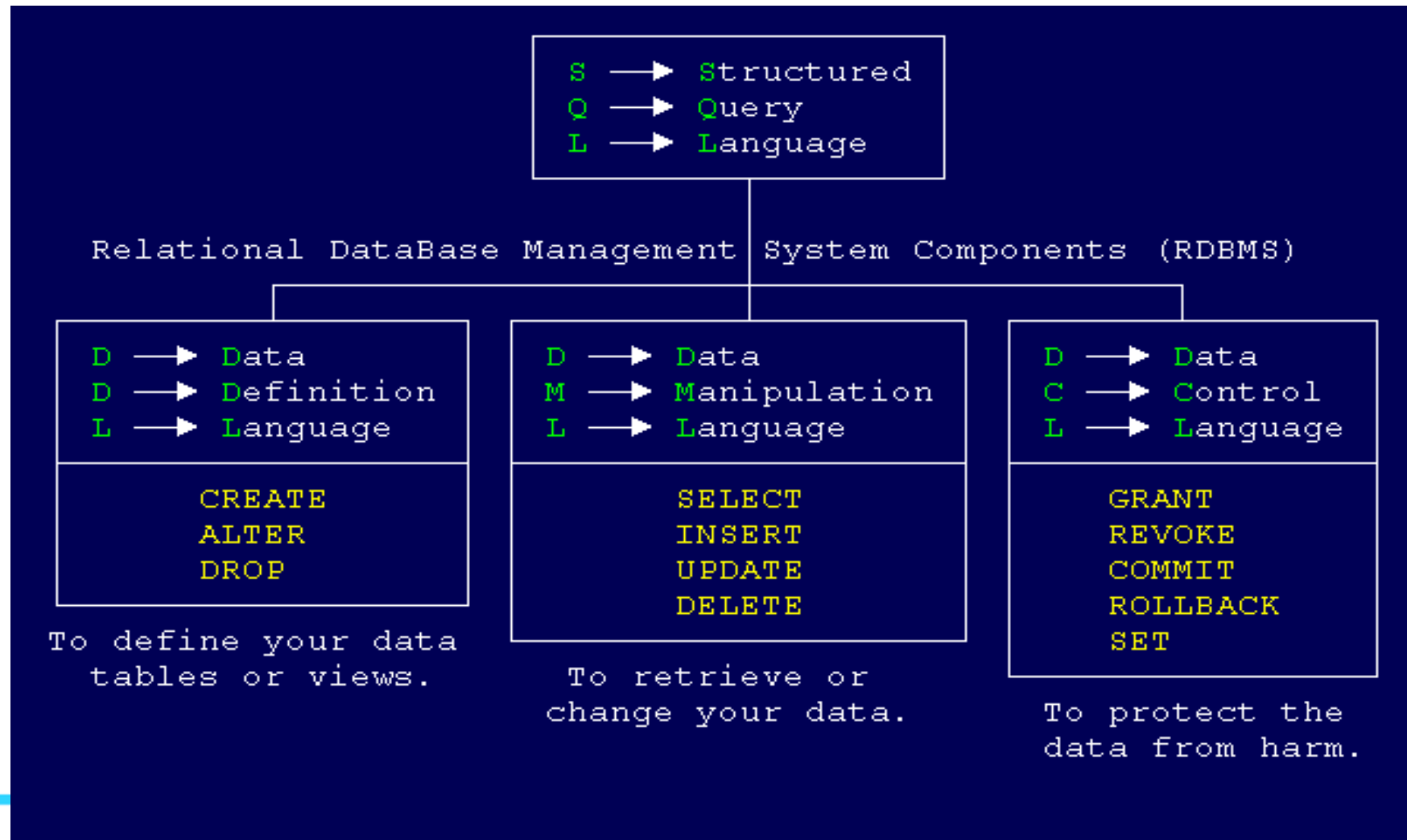
Banco de Dados

SQL - Histórico

- Desenvolvido originalmente nos anos 70 (74-79) nos laboratórios da IBM em San Jose projeto de SGBD chamado System R
- Objetivava demonstrar a viabilidade do modelo relacional proposto por E.F.Codd
- Linguagem padrão para bancos de dados
- SQL significa **Structured Query Language** ou linguagem estruturada de CONSULTA.
 - porém é mais que consulta
 - inclusão, alteração e exclusão

Banco de Dados

SQL - Componentes (DDL, DML e DCL)



Banco de Dados

SQL - Componentes (DDL, DML e DCL)

- Uma **DDL** para definição do esquema da base de dados
- Uma **DML** para programação de consultas e transações que inserem, removem e alteram linhas de tabelas
- Uma **DCL** controla os aspectos de autorização de dados e licenças de usuários para controlar quem tem acesso para ver ou manipular dados dentro do banco de dados.

Banco de Dados

SQL - Componentes (DDL)

- Criar e modificar tabelas
- **Definir** views
- **Definir** índices
- **Definir** e controlar formas de armazenamento físico

Banco de Dados

SQL - Componentes (DDL)

Instruções da DDL

SQL oferece três instruções para definição do esquema da base de dados:

- **Create Table** (define a estrutura de uma tabela, suas restrições de integridade e cria a tabela vazia)
- **Drop Table** (elimina a tabela da base de dados)
- **Alter Table** (permite modificar a definição de uma tabela)

Banco de Dados

DDL - Create table

departamento (**coddepto**, nomdepto)

funcionario (**codfunc**, nomfunc, datanasc, cpf,
#**coddepto**, salario)

Chave estrangeira

coddepto referencia departamento (coddepto)

Banco de Dados

DDL - Create table

```
create table departamento  
    (coddepto integer,  
    nomdepto varchar(40),  
    primary key (coddepto));
```

```
create table funcionario  
    (codfunc integer,  
    nomfunc varchar(40),  
    datanasc date,  
    cpf integer unique,  
    coddepto integer not null,  
    salario numeric (12,2),  
    primary key (codfunc),  
    foreign key (coddepto) references departamento (coddepto);
```


Banco de Dados

DDL - Create table

Exercício sobre criação de tabelas:

Digitar os comandos SQL dos exercícios abaixo em um editor de textos.

- **Vinhos e pesquisa (1 e 2)**

Banco de Dados

DML

Linguagem de Manipulação de Dados (ou DML, de Data Manipulation Language)
Porção da linguagem SQL que possui comandos para “manipulação dos dados”.

As DMLs têm sua capacidade funcional organizada pela palavra inicial em uma declaração, a qual é quase sempre um verbo. No caso da SQL-DML, estes verbos são:

- Select (selecionar/consultar dados)
- Insert (inserir)
- Update (atualizar)
- Delete (apagar ou “deletar”)

Banco de Dados

Insert

Função: incluir informações em uma tabela

Sintaxe:

- insert into nome_tabela (lista-de-campos) values (lista_dados)
- ou
- insert into nome_tabela values (lista_dados)

Exemplos:

- insert into funcionarios (codigo, nome) values ('1','José');
- insert into funcionarios values ('1','José');
- insert into funcionarios (nome) values ('José');

Banco de Dados

Insert

Insert em campo do tipo “Serial”

```
create table funcionarios (  
  id serial,  
  nome varchar(40),  
  primary key(id));  
  
insert into funcionarios (nome) values ('José');  
insert into funcionarios (nome) values ('Maria');  
insert into funcionarios (nome) values ('Ana');  
insert into funcionarios (nome) values ('Pedro');  
  
Select * from funcionarios;
```

Output pane

Data Output Explain Messages History

	Id Integer	nome character varying(40)
1	1	José
2	2	Maria
3	3	Ana
4	4	Pedro

Banco de Dados

Insert

Múltiplos Inserts

```
insert into funcionarios (nome,depto_id,cidade_id)
values ('Maria da Silva',1,2),('Pedro da Silva',2,1),('Ana da Silva',3,2),('João da Silva',2,1);
```

Output pane

Data Output Explain **Messages** History

Query returned successfully: 4 rows affected, 41 ms execution time.

```
insert into funcionarios (nome,depto_id,cidade_id)
values ('Maria da Silva',1,2);
insert into funcionarios (nome,depto_id,cidade_id)
values ('Pedro da Silva',2,1);
insert into funcionarios (nome,depto_id,cidade_id)
values ('Ana da Silva',3,2);
insert into funcionarios (nome,depto_id,cidade_id)
values ('João da Silva',2,1);
```

Output pane

Data Output Explain **Messages** History

Query returned successfully: one row affected, 41 ms execution time.

Banco de Dados

Update

Função: alterar as informações de uma tabela

Sintaxe:

```
update nome_tabela set campo = 'novo_valor' where condição
```

Exemplos:

```
update funcionarios set nome='joão' where id=1;
```

```
update funcionarios set nome='josé' where nome='joão';
```

```
update funcionarios set nome= (concat(nome,' Silva')) where id=1;
```

```
update funcionarios set nome='josé',email='jose@gmail.com' where id='1';
```

```
update funcionarios set salario=salario*1.1 where depto_id='1';
```

```
update funcionarios set salario=salario*1.1 where depto_id='1' and salario < 1000;
```

```
update funcionarios set salario=salario*0.9;
```

Banco de Dados

Delete

Função: apagar as informações de uma tabela

Sintaxe:

`delete from nome_tabela where condição`

Exemplos:

`delete from funcionarios;`

`delete from funcionarios where id=5;`

`delete from funcionarios where depto_id = 3 and salario > 10000;`

Banco de Dados

Select

Função: consultar as informações de uma tabela

Sintaxe:

```
select campo1,campo2 from table1 where condição;
```

ou

```
select campo1,campo2  
from table1  
where condição;
```

Exemplos:

```
select id, nome from funcionarios;  
select id,nome,email from funcionarios where id=3;  
select id,nome,email from funcionarios where id>5;  
select id,nome,email from funcionarios;  
select * from funcionarios;  
select * from funcionarios where depto_id=5;
```


Banco de Dados

upper / lower

upper / lower - converte para maiúsculas / minúsculas

Ex:

```
select upper(nome) from funcionarios;  
select lower(nome) from funcionarios;  
select * from funcionarios where UPPER(nome)='JOSÉ' ;
```

like - permite a utilização do % (coringa)

Ex:

```
select * from funcionarios where nome like 'José%';  
select * from funcionarios where nome like 'A%';  
select * from funcionarios where nome like '%Silva';  
select * from funcionarios where nome like '%Silva%';
```

Banco de Dados

Select

Função: consultar as informações de uma tabela (**ordenação**)

Sintaxe:

```
select campo1,campo2 from table1 where condição;
```

```
select campo1,campo2  
  from table1  
  where condição  
  order by campo;
```

Exemplos:

```
select * from funcionarios order by id;  
select id,nome,email from funcionarios order by nome;  
select id,nome,email from funcionarios order by nome desc;  
select id,nome,email from funcionarios order by id desc;
```

Banco de Dados

Select - informação de várias tabelas

Função: consultar as informações de mais de uma tabela (**junção de tabelas**)

O comando select permite juntar (junção) de informações de duas ou mais tabelas.

Sintaxe:

```
select table1.campo1,table2.campo2
      from table1 , table2
      where condição (aqui deve ser especifica a junção das tabelas)
```

Exemplos:

funcionarios (id, nome, depto_id)

departamento (id, descricao)

```
select funcionarios.nome,departamento.descricao
      from funcionarios,departamentos
      where funcionarios.depto_id = departamentos.id
```

Banco de Dados

Select - informação de várias tabelas

Exemplos - utilização de “alias” ou “apelido”:

funcionarios (id, nome, depto_id)

departamentos (id, descricao)

select f.nome,d.descricao

from funcionarios **as** f,departamentos **as** d

where f.depto_id = d.id

Banco de Dados

Select - informação de várias tabelas

Exemplos - utilização de “alias” ou “apelido”:

funcionarios (id, nome, depto_id, cidade_id)

departamentos (id, descricao)

cidades (id, nome)

select f.nome, d.descricao

from funcionarios as f, departamentos as d, cidades as c

where f.depto_id = d.id and f.cidade_id=c.id

Banco de Dados

Select - informação de várias tabelas

```
create table cidades (  
id serial,  
  nome varchar(40),  
primary key(id));
```

```
create table departamentos (  
id serial,  
  descricao varchar(30),  
primary key(id));
```

Banco de Dados

Select - informação de várias tabelas

```
create table funcionarios (  
    id serial,  
    nome varchar(40),  
    depto_id int,  
    cidade_id int,  
    primary key(id),  
    foreign key (depto_id) references departamentos(id),  
    foreign key (cidade_id) references cidades(id)  
);
```

Banco de Dados

Select - informação de várias tabelas

Cidades

	Id [PK] serial	nome character varying(40)
1	1	Pelotas
2	2	Porto Alegre
3	3	São Paulo
4	4	Rio de Janeiro
*		

Departamentos

	Id [PK] serial	descricao character varying(30)
1	1	Financeiro
2	2	Contábil
3	3	TI
*		

Funcionários

	Id [PK] serial	nome character v	depto_id Integer	cidade_id Integer
1	1	José	1	1
2	2	Maria	2	3
3	3	Pedro	1	4
4	4	Ana	2	1
*				

Banco de Dados

Select - informação de várias tabelas

Comando select para visualizar:

nome de todos os funcionários

nome das suas respectivas cidade e seus departamentos

Banco de Dados

Select - informação de várias tabelas

Comando select para visualizar:

nome de todos os funcionários

nome das suas respectivas cidade e seus departamentos

Solução:

select f.nome, c.nome, d.descricao

from funcionarios as f, cidades as c, departamentos as d

Banco de Dados

Select - informação de várias tabelas

Resultado:

	nome character varying(40)	nome character varying(40)	descricao character varying(30)
1	José	Pelotas	Financeiro
2	José	Pelotas	Contábil
3	José	Pelotas	TI
4	Maria	Pelotas	Financeiro
5	Maria	Pelotas	Contábil
6	Maria	Pelotas	TI
7	Pedro	Pelotas	Financeiro
8	Pedro	Pelotas	Contábil
9	Pedro	Pelotas	TI
10	Ana	Pelotas	Financeiro
11	Ana	Pelotas	Contábil
12	Ana	Pelotas	TI
13	José	Porto Alegre	Financeiro
14	José	Porto Alegre	Contábil
15	José	Porto Alegre	TI
16	Maria	Porto Alegre	Financeiro
17	Maria	Porto Alegre	Contábil
18	Maria	Porto Alegre	TI
19	Pedro	Porto Alegre	Financeiro
20	Pedro	Porto Alegre	Contábil
21	Pedro	Porto Alegre	TI

22	Ana	Porto Alegre	Financeiro
23	Ana	Porto Alegre	Contábil
24	Ana	Porto Alegre	TI
25	José	São Paulo	Financeiro
26	José	São Paulo	Contábil
27	José	São Paulo	TI
28	Maria	São Paulo	Financeiro
29	Maria	São Paulo	Contábil
30	Maria	São Paulo	TI
31	Pedro	São Paulo	Financeiro
32	Pedro	São Paulo	Contábil
33	Pedro	São Paulo	TI
34	Ana	São Paulo	Financeiro
35	Ana	São Paulo	Contábil
36	Ana	São Paulo	TI
37	José	Rio de Janeiro	Financeiro
38	José	Rio de Janeiro	Contábil
39	José	Rio de Janeiro	TI
40	Maria	Rio de Janeiro	Financeiro
41	Maria	Rio de Janeiro	Contábil
42	Maria	Rio de Janeiro	TI
43	Pedro	Rio de Janeiro	Financeiro
44	Pedro	Rio de Janeiro	Contábil
45	Pedro	Rio de Janeiro	TI
46	Ana	Rio de Janeiro	Financeiro
47	Ana	Rio de Janeiro	Contábil
48	Ana	Rio de Janeiro	TI

Banco de Dados

Select - informação de várias tabelas

O resultado é o produto cartesiano de:

4 cidades X 3 departamentos X 4 funcionários = 48 registros

Banco de Dados

Select - informação de várias tabelas

Quando o select envolve mais de uma tabela é **SEMPRE** necessário indicar os pontos de junção (PK = FK) na cláusula **WHERE** do comando **SELECT**.

Banco de Dados

Select - informação de várias tabelas

Quando o select envolve mais de uma tabela é **SEMPRE** necessário indicar os pontos de junção (PK = FK) na cláusula **WHERE** do comando **SELECT**.

```
select f.nome,c.nome,d.descricao  
from funcionarios as f, cidades as c, departamentos as d  
where f.cidade_id = c.id and f.depto_id = d.id;
```

Output pane

Data Output

Explain

Messages

History

	nome character varying(40)	nome character varying(40)	descricao character varying(30)
1	José	Pelotas	Financeiro
2	Maria	São Paulo	Contábil
3	Pedro	Rio de Janeiro	Financeiro
4	Ana	Pelotas	Contábil

Banco de Dados

Select - informação de várias tabelas

```
select f.nome,c.nome,d.descricao  
from funcionarios as f, cidades as c, departamentos as d  
where f.cidade_id = c.id and f.depto_id = d.id and c.nome like 'P%';
```

put pane

Data Output

Explain

Messages

History

	nome character varying(40)	nome character varying(40)	descricao character varying(30)
1	José	Pelotas	Financeiro
2	Ana	Pelotas	Contábil

Banco de Dados

Select - informação de várias tabelas

```
select f.nome,c.nome,d.descricao  
from funcionarios as f, cidades as c, departamentos as d  
where f.cidade_id = c.id and f.depto_id = d.id and d.id = 1;
```

Output pane

	nome character varying(40)	nome character varying(40)	descricao character varying(30)
1	José	Pelotas	Financeiro
2	Pedro	Rio de Janeiro	Financeiro

Banco de Dados

Select - informação de várias tabelas

```
select f.nome,c.nome,d.id,d.descricao  
from funcionarios as f, cidades as c, departamentos as d  
where f.cidade_id = c.id and f.depto_id = d.id and d.id = 1;
```

Output pane

Data Output

Explain

Messages

History

	nome character varying(40)	nome character varying(40)	Id Integer	descricao character varying(30)
1	José	Pelotas	1	Financeiro
2	Pedro	Rio de Janeiro	1	Financeiro

Banco de Dados

Select - informação de várias tabelas

```
select f.nome as "Nome do Funcionário",c.nome as "Nome da Cidade",d.id as "Cód.do Departamento",d.descricao as "Nome do Setor"  
from funcionarios as f, cidades as c, departamentos as d  
where f.cidade_id = c.id and f.depto_id = d.id and d.id = 1;
```

Output pane

Data Output

Explain

Messages

History

	Nome do Funcionário character varying(40)	Nome da Cidade character varying(40)	Cód.do Departamento Integer	Nome do Setor character varying(30)
1	José	Pelotas	1	Financeiro
2	Pedro	Rio de Janeiro	1	Financeiro

Banco de Dados

Select - informação de várias tabelas

Nome do Funcionário	Nome da Cidade	Cód.do Departamento	Nome do Setor
José	Pelotas	1	Financeiro
Pedro	Rio de Janeiro	1	Financeiro

(2 rows)

Banco de Dados

Comando Join

Em alguns casos a junção entre tabelas não deve ser construída utilizando apenas o comando **SELECT**. Por exemplo:

Tabela:funcionarios

	Id [PK] serial	nome character varying(40)	depto_id Integer	cidade_id Integer
1	1	José Santos	1	1
2	2	Maria Santos	2	3
3	3	Pedro Santos	1	4
4	4	Ana Santos	2	1
5	5	José Souza	1	
6	6	Ana Souza	2	
7	7	Maria da Silva	1	2
8	8	Pedro da Silva	2	1
9	9	Ana da Silva		2
10	10	João da Silva		1
*				

Tabela: cidades

Id [PK] serial	nome character varying(40)
1	Pelotas
2	Porto Alegre
3	São Paulo
4	Rio de Janeiro

Tabela: departamentos

	Id [PK] serial	descricao character varying(30)
1	1	Financeiro
2	2	Contábil
3	3	TI
*		

Banco de Dados

Comando Join

Select para relacionar as duas tabelas:

```
select * from funcionarios as f, cidades as c  
where f.cidade_id = c.id
```

Output pane

Data Output		Explain	Messages	History		
	Id Integer	nome character varying(40)	depto_Id Integer	cidade_Id Integer	Id Integer	nome character varying(40)
1	7	Maria da Silva	1	2	2	Porto Alegre
2	8	Pedro da Silva	2	1	1	Pelotas
3	1	José Santos	1	1	1	Pelotas
4	2	Maria Santos	2	3	3	São Paulo
5	3	Pedro Santos	1	4	4	Rio de Janeiro
6	4	Ana Santos	2	1	1	Pelotas
7	10	João da Silva		1	1	Pelotas
8	9	Ana da Silva		2	2	Porto Alegre

Banco de Dados

Comando Join

Select para relacionar as duas tabelas:

previous queries

```
select * from funcionarios as f,departamentos as d
where f.depto_id = d.id
```

Output pane

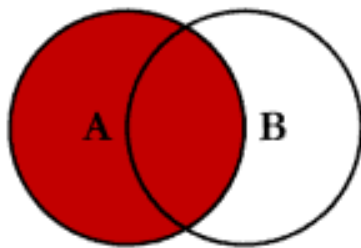
Data Output Explain Messages History

	Id Integer	nome character varying(40)	depto_id Integer	cidade_id Integer	Id Integer	descricao character varying(30)
1	7	Maria da Silva	1	2	1	Financeiro
2	8	Pedro da Silva	2	1	2	Contábil
3	5	José Souza	1		1	Financeiro
4	6	Ana Souza	2		2	Contábil
5	1	José Santos	1	1	1	Financeiro
6	2	Maria Santos	2	3	2	Contábil
7	3	Pedro Santos	1	4	1	Financeiro
8	4	Ana Santos	2	1	2	Contábil

Banco de Dados

Comando Join

LEFT JOIN



Esta consulta retorna todos os registros da tabela esquerda (tabela A) e as correspondências que existirem com a tabela direita (tabela B). O código ficará da seguinte forma:

```
SELECT *  
FROM A  
LEFT JOIN B  
ON A.Key = B.Key
```

Banco de Dados

Comando Join

```
select * from funcionarios as f  
left join cidades as c  
on f.cidade_id = c.id
```

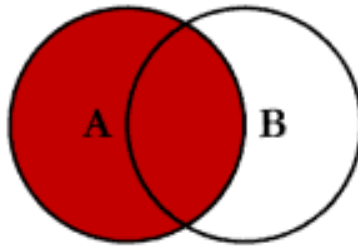
Output pane

Data Output		Explain	Messages	History		
	Id Integer	nome character varying(40)	depto_id Integer	cidade_id Integer	Id Integer	nome character varying(40)
1	7	Maria da Silva	1	2	2	Porto Alegre
2	8	Pedro da Silva	2	1	1	Pelotas
3	5	José Souza	1			
4	6	Ana Souza	2			
5	1	José Santos	1	1	1	Pelotas
6	2	Maria Santos	2	3	3	São Paulo
7	3	Pedro Santos	1	4	4	Rio de Janeiro
8	4	Ana Santos	2	1	1	Pelotas
9	10	João da Silva		1	1	Pelotas
10	9	Ana da Silva		2	2	Porto Alegre

Banco de Dados

Comando Join

LEFT JOIN



Esta consulta retorna todos os registros da tabela esquerda (tabela A) e as correspondências que existirem com a tabela direita (tabela B). O código ficará da seguinte forma:

```
SELECT *  
FROM A  
LEFT JOIN B  
ON A.Key = B.Key
```

Banco de Dados

Comando Join

```
select * from funcionarios as f  
left join departamentos as d  
on f.depto_id = d.id
```

Output pane

Data Output

Explain

Messages

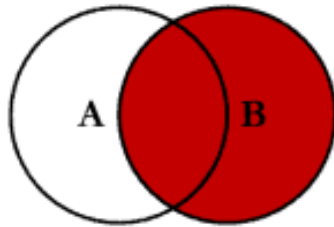
History

	Id Integer	nome character varying(40)	depto_id Integer	cidade_id Integer	Id Integer	descricao character varying(30)
1	7	Maria da Silva	1	2	1	Financeiro
2	8	Pedro da Silva	2	1	2	Contábil
3	5	José Souza	1		1	Financeiro
4	6	Ana Souza	2		2	Contábil
5	1	José Santos	1	1	1	Financeiro
6	2	Maria Santos	2	3	2	Contábil
7	3	Pedro Santos	1	4	1	Financeiro
8	4	Ana Santos	2	1	2	Contábil
9	10	João da Silva		1		
10	9	Ana da Silva		2		

Banco de Dados

Comando Join

RIGHT JOIN



Esta consulta retornará todos os registros da tabela direita (tabela B) e as correspondências que existirem com a tabela esquerda (tabela A). O código ficará da seguinte forma:

```
SELECT *  
FROM A  
RIGHT JOIN B  
ON A.Key = B.Key
```

Banco de Dados

Comando Join

```
select * from funcionarios as f  
right join departamentos as d  
on f.depto_id = d.id
```

Output pane

Data Output

Explain

Messages

History

	Id Integer	nome character varying(40)	depto_id Integer	cidade_id Integer	Id Integer	descricao character varying(30)
1	7	Maria da Silva	1	2	1	Financeiro
2	8	Pedro da Silva	2	1	2	Contábil
3	5	José Souza	1		1	Financeiro
4	6	Ana Souza	2		2	Contábil
5	1	José Santos	1	1	1	Financeiro
6	2	Maria Santos	2	3	2	Contábil
7	3	Pedro Santos	1	4	1	Financeiro
8	4	Ana Santos	2	1	2	Contábil
9					3	TI

Banco de Dados

Comando Join

```
select * from funcionarios as f  
right join cidades as c  
on f.cidade_id = c.id
```

Output pane

Data Output

Explain

Messages

History

	Id Integer	nome character varying(40)	depto_id Integer	cidade_id Integer	Id Integer	nome character varying(40)
1	7	Maria da Silva	1	2	2	Porto Alegre
2	8	Pedro da Silva	2	1	1	Pelotas
3	1	José Santos	1	1	1	Pelotas
4	2	Maria Santos	2	3	3	São Paulo
5	3	Pedro Santos	1	4	4	Rio de Janeiro
6	4	Ana Santos	2	1	1	Pelotas
7	10	João da Silva		1	1	Pelotas
8	9	Ana da Silva		2	2	Porto Alegre

Banco de Dados

Comando Join

Duas cidades foram incluídas e não foram relacionadas com funcionários (Curitiba e Brasília)

```
select * from funcionarios as f  
right join cidades as c  
on f.cidade_id = c.id
```

Output pane

	Data Output	Explain	Messages	History		
	Id Integer	nome character varying(40)	depto_id Integer	cidade_id Integer	Id Integer	nome character varying(40)
1	7	Maria da Silva	1	2	2	Porto Alegre
2	8	Pedro da Silva	2	1	1	Pelotas
3	1	José Santos	1	1	1	Pelotas
4	2	Maria Santos	2	3	3	São Paulo
5	3	Pedro Santos	1	4	4	Rio de Janeiro
6	4	Ana Santos	2	1	1	Pelotas
7	10	João da Silva		1	1	Pelotas
8	9	Ana da Silva		2	2	Porto Alegre
9					10	Curitiba
10					9	Brasília

Banco de Dados

Comando Join

Left Join com 3 tabelas

```
select * from funcionarios as f
left join departamentos as d
on f.depto_id = d.id
left join cidades as c
on f.cidade_id = c.id
```

Output pane

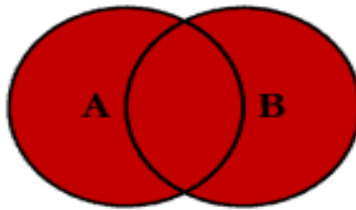
Data Output		Explain	Messages	History				
	Id Integer	nome character varying(40)	depto_id Integer	cidade_id Integer	Id Integer	descricao character varying(30)	Id Integer	nome character varying(40)
1	7	Maria da Silva	1	2	1	Financeiro	2	Porto Alegre
2	8	Pedro da Silva	2	1	2	Contábil	1	Pelotas
3	5	José Souza	1		1	Financeiro		
4	6	Ana Souza	2		2	Contábil		
5	1	José Santos	1	1	1	Financeiro	1	Pelotas
6	2	Maria Santos	2	3	2	Contábil	3	São Paulo
7	3	Pedro Santos	1	4	1	Financeiro	4	Rio de Janeiro
8	4	Ana Santos	2	1	2	Contábil	1	Pelotas
9	10	João da Silva		1			1	Pelotas
10	9	Ana da Silva		2			2	Porto Alegre

Banco de Dados

Comando Join

Full Outer Join ou Full Join

OUTER JOIN



Este JOIN também é conhecido como FULL OUTER JOIN ou FULL JOIN. Esta consulta retornará todos os registros das duas tabelas e juntando também os registros correspondentes entre as duas tabelas. O código ficará da seguinte forma:

```
SELECT *  
FROM A  
FULL OUTER JOIN B B  
ON A.Key = B.Key
```

Banco de Dados

Comando Join

Full Outer Join ou Full Join

```
select * from funcionarios as f
full outer join departamentos as d
on f.depto_id = d.id
```

Output pane

	Id Integer	nome character varying(40)	depto_id Integer	cidade_id Integer	Id Integer	descricao character varying(30)
3	5	José Souza	1		1	Financeiro
4	6	Ana Souza	2		2	Contábil
5	1	José Santos	1	1	1	Financeiro
6	2	Maria Santos	2	3	2	Contábil
7	3	Pedro Santos	1	4	1	Financeiro
8	4	Ana Santos	2	1	2	Contábil
9	10	João da Silva		1		
10	9	Ana da Silva		2		
11					5	Marketing
12					4	Vendas
13					3	TI

Banco de Dados

Comando Join

Full Outer Join ou Full Join

```
select * from funcionarios as f  
full join cidades as c  
on f.cidade_id = c.id
```

Output pane

Data Output		Explain	Messages	History		
	Id Integer	nome character varying(40)	depto_id Integer	cidade_id Integer	Id Integer	nome character varying(40)
1	7	Maria da Silva	1	2	2	Porto Alegre
2	8	Pedro da Silva	2	1	1	Pelotas
3	5	José Souza	1			
4	6	Ana Souza	2			
5	1	José Santos	1	1	1	Pelotas
6	2	Maria Santos	2	3	3	São Paulo
7	3	Pedro Santos	1	4	4	Rio de Janeiro
8	4	Ana Santos	2	1	1	Pelotas
9	10	João da Silva		1	1	Pelotas
10	9	Ana da Silva		2	2	Porto Alegre
11					10	Curitiba
12					9	Brasília

Banco de Dados

Comando Join

Full Outer Join ou Full Join - 3 tabelas

```
select * from funcionarios as f
full join cidades as c
on f.cidade_id = c.id
full join departamentos as d
on f.depto_id = d.id
```

Output pane

Data Output

Explain

Messages

History

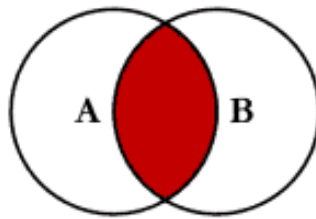
	Id Integer	nome character varying(40)	depto_id Integer	cidade_id Integer	Id Integer	nome character varying(40)	Id Integer	descricao character varying(30)
1	7	Maria da Silva	1	2	2	Porto Alegre	1	Financeiro
2	8	Pedro da Silva	2	1	1	Pelotas	2	Contábil
3	5	José Souza	1				1	Financeiro
4	6	Ana Souza	2				2	Contábil
5	1	José Santos	1	1	1	Pelotas	1	Financeiro
6	2	Maria Santos	2	3	3	São Paulo	2	Contábil
7	3	Pedro Santos	1	4	4	Rio de Janeiro	1	Financeiro
8	4	Ana Santos	2	1	1	Pelotas	2	Contábil
9	10	João da Silva		1	1	Pelotas		
10	9	Ana da Silva		2	2	Porto Alegre		
11					10	Curitiba		
12					9	Brasília		
13							5	Marketing
14							4	Vendas
15							3	TI

Banco de Dados

Comando Join

Inner Join

INNER JOIN



Este é simples, o mais entendível e o mais comum. Esta consulta retornará todos os registros da tabela esquerda (tabela A) que têm correspondência com a tabela direita (tabela B). Podemos escrever este JOIN da seguinte forma:

```
SELECT *  
FROM A  
INNER JOIN B  
ON A.Key = B.Key
```

Banco de Dados

Comando Join

Inner Join

```
select * from funcionarios as f  
inner join cidades as c  
on f.cidade_id = c.id
```

Output pane

Data Output

Explain

Messages

History

	Id Integer	nome character varying(40)	depto_id Integer	cidade_id Integer	Id Integer	nome character varying(40)
1	7	Maria da Silva	1	2	2	Porto Alegre
2	8	Pedro da Silva	2	1	1	Pelotas
3	1	José Santos	1	1	1	Pelotas
4	2	Maria Santos	2	3	3	São Paulo
5	3	Pedro Santos	1	4	4	Rio de Janeiro
6	4	Ana Santos	2	1	1	Pelotas
7	10	João da Silva		1	1	Pelotas
8	9	Ana da Silva		2	2	Porto Alegre

Banco de Dados

Comando Join

Inner Join

```
select * from funcionarios as f
inner join cidades as c
on f.cidade_id = c.id
inner join departamentos as d
on f.depto_id = d.id
```

Output pane

Data Output

Explain

Messages

History

	Id Integer	nome character varying(40)	depto_id Integer	cidade_id Integer	Id Integer	nome character varying(40)	Id Integer	descricao character varying(30)
1	7	Maria da Silva	1	2	2	Porto Alegre	1	Financeiro
2	8	Pedro da Silva	2	1	1	Pelotas	2	Contábil
3	1	José Santos	1	1	1	Pelotas	1	Financeiro
4	2	Maria Santos	2	3	3	São Paulo	2	Contábil
5	3	Pedro Santos	1	4	4	Rio de Janeiro	1	Financeiro
6	4	Ana Santos	2	1	1	Pelotas	2	Contábil

Banco de Dados

Comando Join

Select “comum” relacionando 3 tabelas

```
select * from funcionarios as f, cidades as c, departamentos as d  
where f.cidade_id = c.id and f.depto_id = d.id
```

Output pane

Data Output

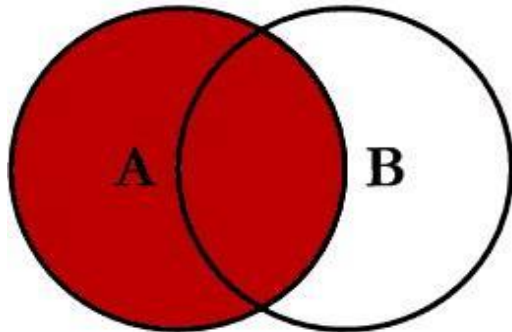
Explain

Messages

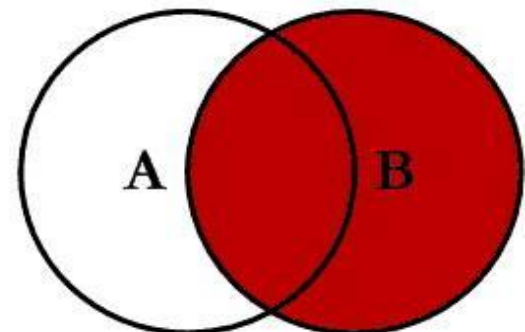
History

	Id Integer	nome character varying(40)	depto_id Integer	cidade_id Integer	Id Integer	nome character varying(40)	Id Integer	descricao character varying(30)
1	7	Maria da Silva	1	2	2	Porto Alegre	1	Financeiro
2	8	Pedro da Silva	2	1	1	Pelotas	2	Contábil
3	1	José Santos	1	1	1	Pelotas	1	Financeiro
4	2	Maria Santos	2	3	3	São Paulo	2	Contábil
5	3	Pedro Santos	1	4	4	Rio de Janeiro	1	Financeiro
6	4	Ana Santos	2	1	1	Pelotas	2	Contábil

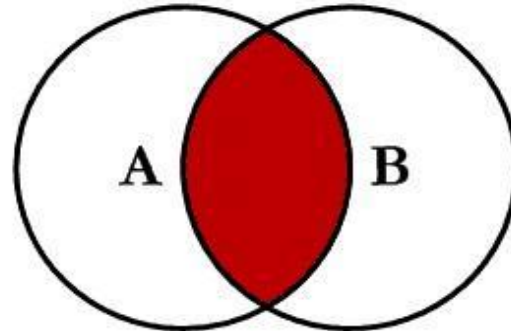
SQL JOINS



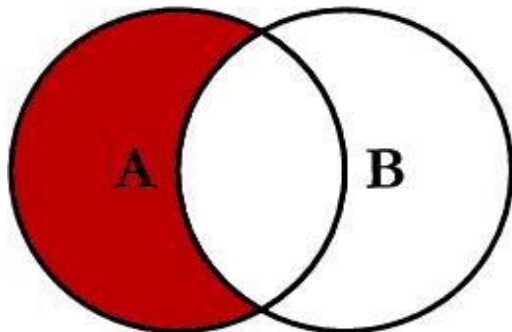
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



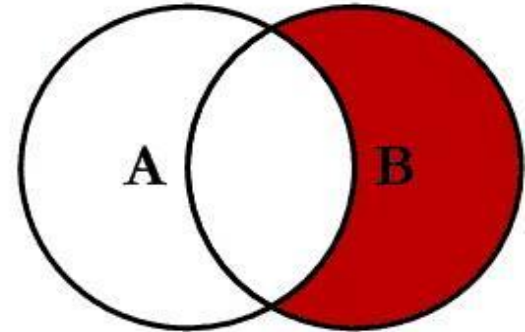
```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```



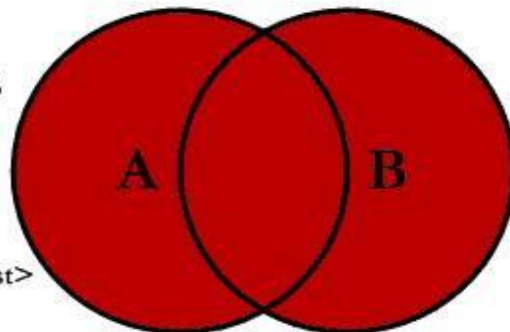
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



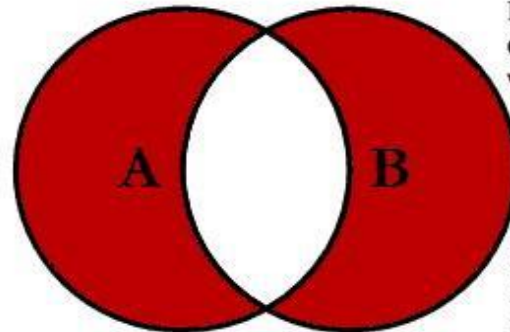
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL  
OR B.Key IS NULL
```

Banco de Dados

Funções

Max - Maior valor

Min - Menor valor

Count - Conta

Sum - Soma

Avg - Média

**Group by
distinct**

Banco de Dados

Funções

Max - Maior valor

Exemplo:

produto (id, descricao, categoria_id, valor)

	Id [PK] serial	descricao character v	categoria_id Integer	valor numeric(6,2)
1	1	Produto 1	1	500.00
2	2	Produto 2	1	700.00
3	3	Produto 3	1	100.00
4	4	Produto 4	1	900.00
5	5	Produto 5	2	100.00
6	6	Produto 6	2	400.00
*				

Banco de Dados

Funções

Max - Maior valor

Exemplo:

produto (id, descricao, categoria_id, valor)

	Id [PK] serial	descricao character v	categoria_id Integer	valor numeric(6,2)
1	1	Produto 1	1	500.00
2	2	Produto 2	1	700.00
3	3	Produto 3	1	100.00
4	4	Produto 4	1	900.00
5	5	Produto 5	2	100.00
6	6	Produto 6	2	400.00
*				

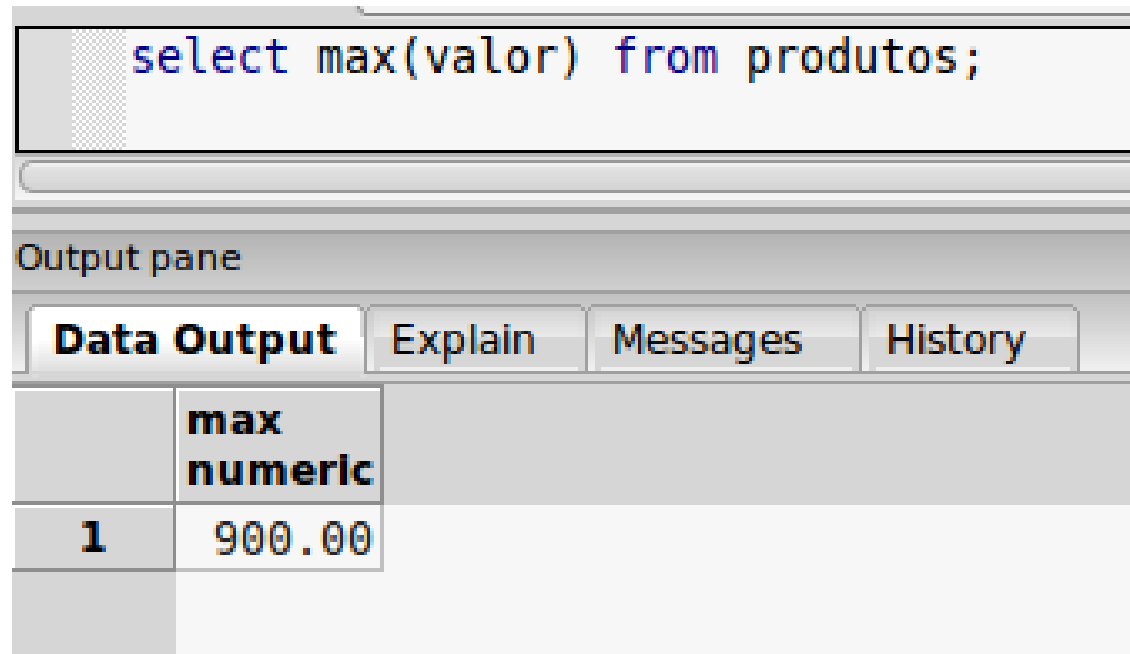
Descobrir o maior valor:

select max(valor) from produtos;

Banco de Dados

Funções

Max - Maior valor



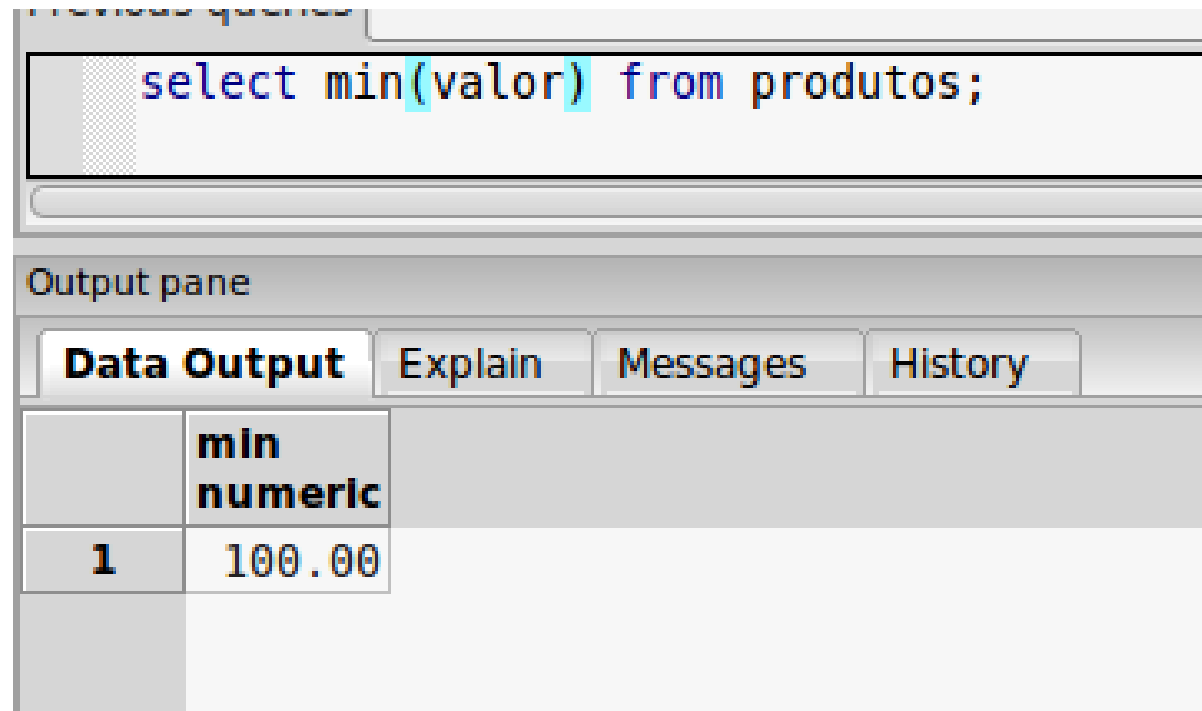
The screenshot shows a database query interface. At the top, a text box contains the SQL query: `select max(valor) from produtos;`. Below this is an "Output pane" with four tabs: "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is selected, displaying a table with the results of the query. The table has two columns: an index column and a column for the maximum value. The first row shows the index "1" and the value "900.00".

	max numeric
1	900.00

Banco de Dados

Funções

Min - Menor valor



The screenshot shows a database query interface. At the top, a text box contains the SQL query: `select min(valor) from produtos;`. Below this 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.

	min numeric
1	100.00

Banco de Dados

Funções

Count - Contar produtos

Previous queries

```
select count(*) from produtos;
```

Output pane

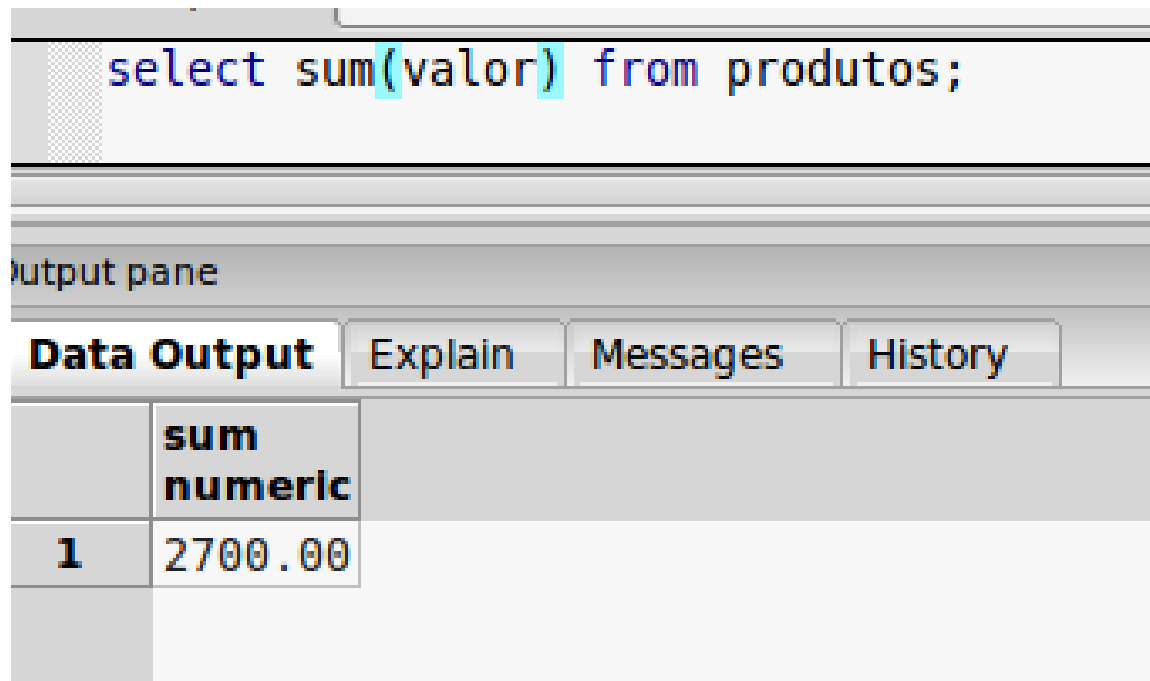
Data Output Explain Messages History

	count bigint
1	6

Banco de Dados

Funções

Sum - Somar o valor dos produtos



The screenshot shows a database query interface. At the top, a text area contains the SQL query: `select sum(valor) from produtos;`. Below this is a section labeled "Output pane" which contains four tabs: "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is selected, displaying a table with the results of the query. The table has two columns: an index column and a column labeled "sum numeric". The first row of data shows the index "1" and the sum value "2700.00".

	sum numeric
1	2700.00

Banco de Dados

Funções

Avg - Média dos valores dos produtos

<pre>select avg(valor) from produtos;</pre>	
Output pane	
Data Output Explain Messages History	
	avg numeric
1	450.0000000000000000

Banco de Dados

Funções

Combinando funções e subconsultas

```
select * from produtos where valor > (select avg(valor) from produtos);
```

Output pane

Data Output

Explain

Messages

History

	Id Integer	descricao character varying(40)	categoria_Id Integer	valor numeric(6,2)
1	1	Produto 1	1	500.00
2	2	Produto 2	1	700.00
3	4	Produto 4	1	900.00

Banco de Dados

Funções

Combinando funções e subconsultas

```
select * from produtos where valor = (select max(valor) from produtos);
```

Output pane

Data Output		Explain	Messages	History
	Id integer	descricao character varying(40)	categoria_id integer	valor numeric(6,2)
1	4	Produto 4	1	900.00

Banco de Dados

Agrupamento (group by)

Criação de agrupamentos com group by e utilização de funções - a cláusula group by agrupa registros usando uma determinada coluna.

No caso do exemplo abaixo, o agrupamento é construído baseado nas informações da coluna categoria_id
Diversas funções são aplicadas ao agrupamento.

```
select categoria_id,max(valor),min(valor),count(*),sum(valor),avg(valor) from produtos group by categoria_id
```

Output pane

Data Output

Explain

Messages

History

	categoria_id Integer	max numeric	min numeric	count bigInt	sum numeric	avg numeric
1	1	900.00	100.00	4	2200.00	550.000000000000000000
2	2	400.00	100.00	2	500.00	250.000000000000000000

Banco de Dados

Agrupamento (group by)

Agrupamento com a cláusula Group by e relacionamento entre tabelas.

```
select c.descricao,max(p.valor),min(p.valor),count(*),sum(p.valor),avg(p.valor)
from produtos as p, categorias as c
where p.categoria_id = c.id
group by c.descricao
```

Output pane

Data Output

Explain

Messages

History

	descricao character varying(40)	max numeric	min numeric	count bigint	sum numeric	avg numeric
1	Categoria 2	400.00	100.00	2	500.00	250.0000
2	Categoria 1	900.00	100.00	4	2200.00	550.0000

Banco de Dados

Agrupamento (group by)

Complementando com having

```
select c.descricao,max(p.valor),min(p.valor),count(*),sum(p.valor),avg(p.valor)
from produtos as p, categorias as c
where p.categoria_id = c.id
group by c.descricao
having sum(valor) = 500
```

Output pane

Data Output

Explain

Messages

History

	descricao character varying(40)	max numeric	min numeric	count bigint	sum numeric	avg numeric
1	Categoria 2	400.00	100.00	2	500.00	250.0000

Banco de Dados

Agrupamento (group by)

Complementando com having

Previous queries

```
select c.descricao,max(p.valor),min(p.valor),count(*),sum(p.valor),avg(p.valor)
from produtos as p, categorias as c
where p.categoria_id = c.id
group by c.descricao
having count(*) < 3
```

Output pane

Data Output Explain Messages History

	descricao character varying(40)	max numeric	min numeric	count bigint	sum numeric	avg numeric	
1	Categoria 2	400.00	100.00	2	500.00	250.0000	

Banco de Dados Distinct

Mostrar sem repetições

**Exemplo: apresentar as categorias que possuem produtos,
sem repetir**

id [PK] serial	nome character varying(40)
1	Categoria 1
2	Categoria 2
3	Categoria 3
4	Categoria 4
5	Categoria 5

id [PK] serial	nome character var	categoria_id integer
1	Produto 1	1
2	Produto 2	1
3	Produto 3	1
4	Produto 4	3
5	Produto 5	3

Banco de Dados Distinct

Mostrar sem repetições

**Exemplo: apresentar as categorias que possuem produtos,
sem repetir**

Previous queries

```
select c.nome from categorias c, produtos p  
where c.id = p.categoria_id
```

Output pane

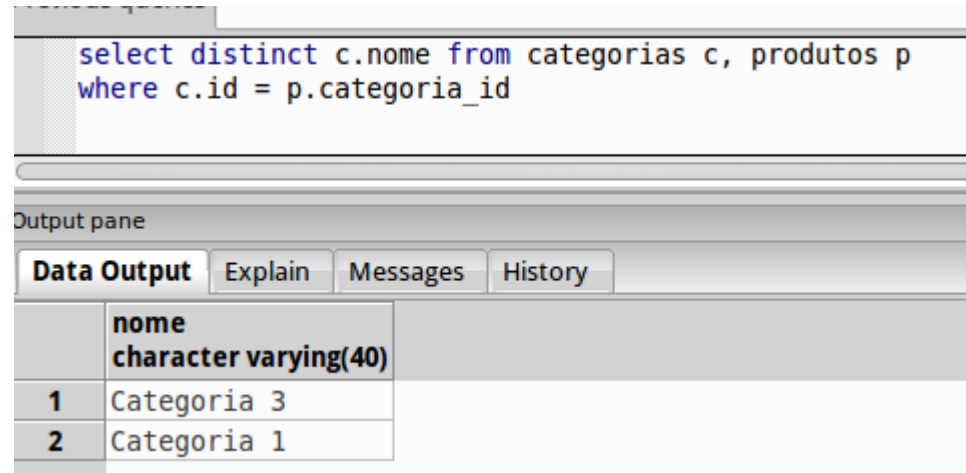
Data Output Explain Messages History

	nome character varying(40)
1	Categoria 1
2	Categoria 1
3	Categoria 1
4	Categoria 3
5	Categoria 3

Banco de Dados Distinct

Mostrar sem repetições

Exemplo: apresentar as categorias que possuem produtos,
sem repetir



The screenshot shows a database query interface. At the top, a SQL query is entered in a text box: `select distinct c.nome from categorias c, produtos p where c.id = p.categoria_id`. Below the query box is an "Output pane" with four tabs: "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is selected, displaying a table with two columns: "nome" (character varying(40)) and an implicit index column. The table contains two rows: "1" for "Categoria 3" and "2" for "Categoria 1".

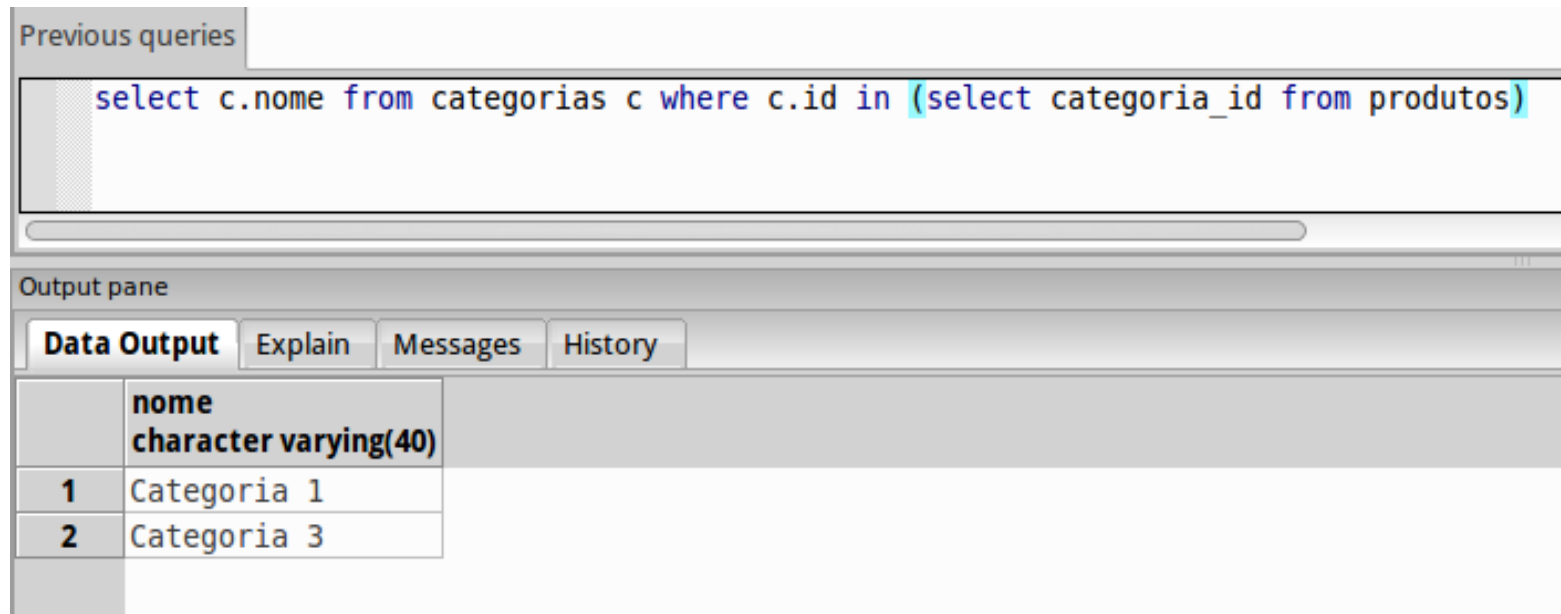
	nome character varying(40)
1	Categoria 3
2	Categoria 1

Banco de Dados

Operador in

Mostrar sem repetições

Exemplo: apresentar as categorias que possuem produtos, sem repetir



The screenshot shows a database query interface. At the top, there is a tab labeled "Previous queries". Below it, a text area contains the SQL query: `select c.nome from categorias c where c.id in (select categoria_id from produtos)`. Below the query area is a horizontal scrollbar. Underneath is the "Output pane" with four tabs: "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is selected, displaying a table with two columns: an index and the category name. The table contains two rows: "1" for "Categoria 1" and "2" for "Categoria 3".

	nome character varying(40)
1	Categoria 1
2	Categoria 3

Banco de Dados Aplicado

Cristiano Santos

cristiano.santos@amf.edu.br