




Banco de dados com PostgreSQL

Aula 3



Uma cláusula JOIN em SQL, correspondente a uma operação de junção em álgebra relacional, combina colunas de uma ou mais tabelas em um banco de dados relacional. Ela cria um conjunto que pode ser salvo como uma tabela ou usado da forma como está.

Um JOIN é um meio de combinar colunas de uma (auto-junção) ou mais tabelas, usando valores comuns a cada uma delas. O SQL padrão ANSI especifica cinco tipos de JOIN: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN e CROSS JOIN.

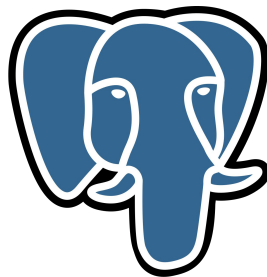
Como um caso especial, uma tabela (tabela base, visão ou tabela juntada) pode se juntar a si mesma em uma auto-união (self-join).

Em um banco de dados relacional, os dados são distribuídos em várias tabelas lógicas. Para obter um conjunto completo e significativo de dados, é necessário consultar dados dessas tabelas usando junções (JOINS).

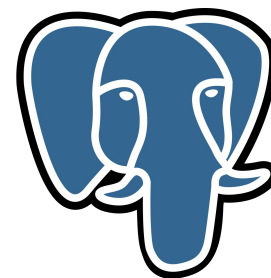
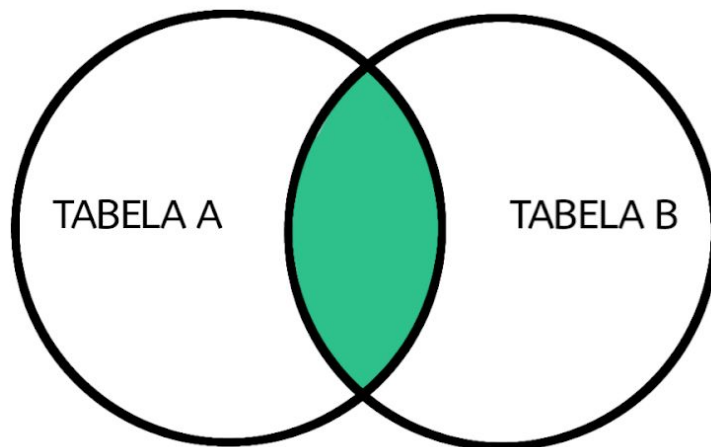


INNER JOIN

A cláusula `INNER JOIN` compara cada linha da tabela A com as linhas da tabela B para encontrar todos os pares de linhas que satisfazem a condição de junção. Se a condição de junção for avaliada como `TRUE`, os valores da coluna das linhas correspondentes das tabelas A e B serão combinados em uma nova linha e incluídos no conjunto de resultados.



```
SELECT *  
FROM RESERVAS R1  
  INNER JOIN PESSOA P1 ON (R1.RESERVADOPOR = P1.ID)  
  INNER JOIN ESPACOLOCACAO E1 ON (R1.ESPACO = E1.ID)
```





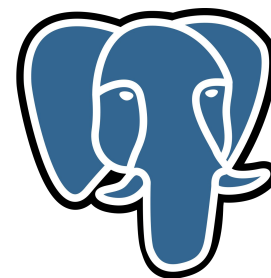
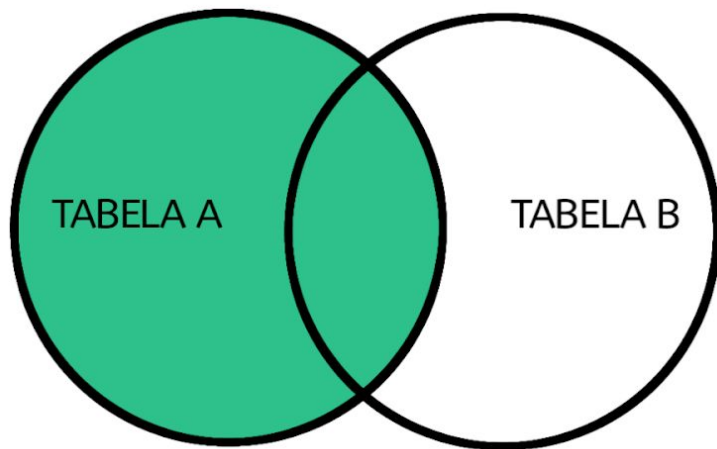
LEFT JOIN

Para cada linha da tabela A, a consulta a compara com todas as linhas da tabela B. Se um par de linhas fizer com que a condição de junção seja avaliado como TRUE, os valores da coluna dessas linhas serão combinados para formar uma nova linha que será incluída no conjunto de resultados.

Se uma linha da tabela “esquerda” A não tiver nenhuma linha correspondente da tabela “direita” B, a consulta irá combinar os valores da coluna da linha da tabela “esquerda” A com NULL para cada valor da coluna da tabela da “direita” B que não satisfaça a condição de junto (FALSE).

Em resumo, a cláusula LEFT JOIN retorna todas as linhas da tabela “esquerda” A e as linhas correspondentes ou valores NULL da tabela “esquerda” A.

```
SELECT *  
FROM RESERVAS R1  
  LEFT JOIN PESSOA P1 ON (R1.RESERVADOPOR = P1.ID)  
  LEFT JOIN ESPACOLOCACAO E1 ON (R1.ESPACO = E1.ID)
```



Exercício 12



Crie uma consulta que liste todas as pessoas cadastradas. Além do nome da pessoa, telefone para contato e e-mail, liste também o nome dos animais vinculados a essa pessoa, além da raça e tipo de animal.

Todas as pessoas cadastradas devem ser exibidas, possuindo animais cadastrados, ou não.

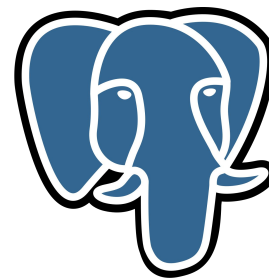




A função COUNT

A função count retorna a quantidade de registros, de acordo com o operador utilizado.

```
SELECT COUNT(*) FROM TB_PRODUTO
```

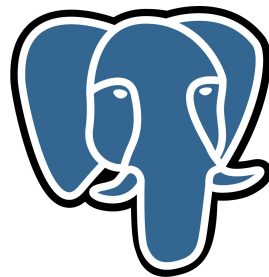




A função COUNT

Utilizando o count com o nome de um campo, contará a quantidade de registros onde esse campo não está nulo.

```
SELECT COUNT (VALOR_PRODUTO) FROM TB_PRODUTO
```

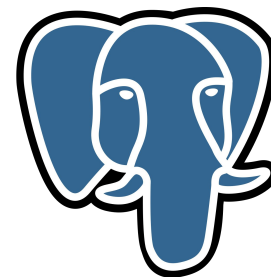




A função COUNT

Utilizando o DISTINCT, somente será contado os valores uma única vez.

```
SELECT COUNT(DISTINCT VALOR_PRODUTO) FROM TB_PRODUTO
```



Exercício 13



Crie um script para contar quantos cachorros estão cadastrados na base de dados.

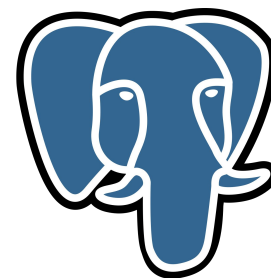




A função SUM

A função SUM retorna a soma de uma coluna numérica.

```
SELECT SUM(V valor_produto) FROM tb_produto
```



Exercício 14

Crie uma tabela chamada `tb_cobranca`. Nessa tabela guardaremos os valores a receber dos clientes em nosso petshop. A tabela de cobrança possuirá os seguintes campos:

`tb_cobranca`

ID

DEVEDOR (Não nulo) (FK de `tb_pessoa`)

VALOR (Não nulo)

VALOR_PAGO

DATA_VENCIMENTO (Não nulo)

DATA_PAGAMENTO

Altere o cadastro da consulta, adicionando um novo campo (`COBRANCA`) que servirá de ligação entre uma cobrança gerada e a consulta. Nem todas as consultas possuirão valor a pagar, nesse caso, o campo de cobrança poderá ser nulo.

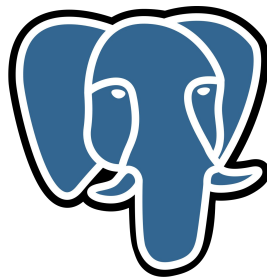
Em seguida, crie uma consulta que exiba somente o valor total a receber.





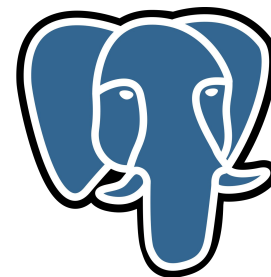
GROUP BY

A cláusula PostgreSQL GROUP BY é usada para dividir as linhas retornadas pela instrução SELECT em grupos diferentes. A especialidade da cláusula GROUP BY é que se pode usar funções como SUM() calcular a soma dos itens ou COUNT() obter o número total de itens nos grupos.



```
SELECT PESSOAS.NOME, SUM(COBRANCAS.VALOR)
FROM COBRANCAS
    INNER JOIN PESSOAS ON (COBRANCAS.DEVEDOR = PESSOAS.ID)
WHERE COBRANCAS.DATAPAGAMENTO IS NULL
GROUP BY PESSOAS.NOME
ORDER BY 2 DESC
```

```
SELECT ESPACOLOCACAO.NOME, COUNT(RESERVAS.ID)
FROM RESERVAS
    INNER JOIN ESPACOLOCACAO ON (RESERVAS.ESPACO =
ESPACOLOCACAO.ID)
GROUP BY ESPACOLOCACAO.NOME
```



Exercício 15



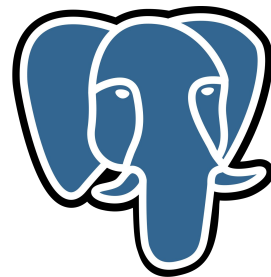
Crie uma consulta, semelhante a criada na aula anterior, que exiba os valores a receber, por pessoa. Somente devem ser exibidas na consulta pessoas que possuam valor a pagar.





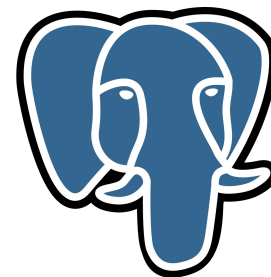
SUBQUERIES

Uma Subquery (também conhecida como SUBCONSULTA ou SUBSELECT) é uma instrução do tipo `SELECT` dentro de outra instrução `SQL`, que efetua consultas que, de outra forma, seriam extremamente complicadas ou impossíveis de serem feitas.



```
SELECT *  
FROM tabela1 AS T  
WHERE coluna1 IN (  
    SELECT coluna2  
    FROM tabela2 AS T2  
    WHERE T.id = T2.id)
```

```
SELECT PESSOAS.NOME,  
       (SELECT SUM(COBRANCAS.VALOR)  
        FROM COBRANCAS  
        WHERE COBRANCAS.DEVEDOR = PESSOAS.ID) AS VALOR_DEVIDO  
FROM PESSOAS
```



Exercício 16



Crie uma consulta para listar os valores a receber de acordo com os tipos de animais. A consulta deve possuir dois campos:

Nome do devedor

Gato: Valor a receber das consultas com gatos

Cachorro: Valor a receber das consultas com cachorros

