



# Consultas Complexas Joins

Herysson R. Figueiredo  
herysson.figueiredo@ufn.edu.br

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	Rua das Flores, 751, São Paulo, SP	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	Rua da Lapa, 34, São Paulo, SP	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	Rua Souza Lima, 35, Curitiba, PR	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	Av. Arthur de Lima, 54, Santo André, SP	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	Rua Rebouças, 65, Piracicaba, SP	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	Av. Lucas Obes, 74, São Paulo, SP	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	Rua Timbira, 35, São Paulo, SP	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	Rua do Horto, 35, São Paulo, SP	M	55.000	NULL	1

DEPARTAMENTO

Dnome	Dnumero	Cpf_gerente	Data_inicio_gerente
Pesquisa	5	33344555587	22-05-1988
Administração	4	98765432168	01-01-1995
Matriz	1	88866555576	19-06-1981

LOCALIZACAO\_DEP

Dnumero	Dlocal
1	São Paulo
4	Mauá
5	Santo André
5	Itu
5	São Paulo

TRABALHA\_EM

Fcpf	Pnr	Horas
12345678966	1	32,5
12345678966	2	7,5
66688444476	3	40,0
45345345376	1	20,0
45345345376	2	20,0
33344555587	2	10,0
33344555587	3	10,0
33344555587	10	10,0
33344555587	20	10,0
99988777767	30	30,0
99988777767	10	10,0
98798798733	10	35,0
98798798733	30	5,0
98765432168	30	20,0
98765432168	20	15,0
88866555576	20	NULL

PROJETO

Projnome	Projnumero	Projlocal	Dnum
ProdutoX	1	Santo André	5
ProdutoY	2	Itu	5
ProdutoZ	3	São Paulo	5
Informatização	10	Mauá	4
Reorganização	20	São Paulo	1
Novosbeneficios	30	Mauá	4

DEPENDENTE

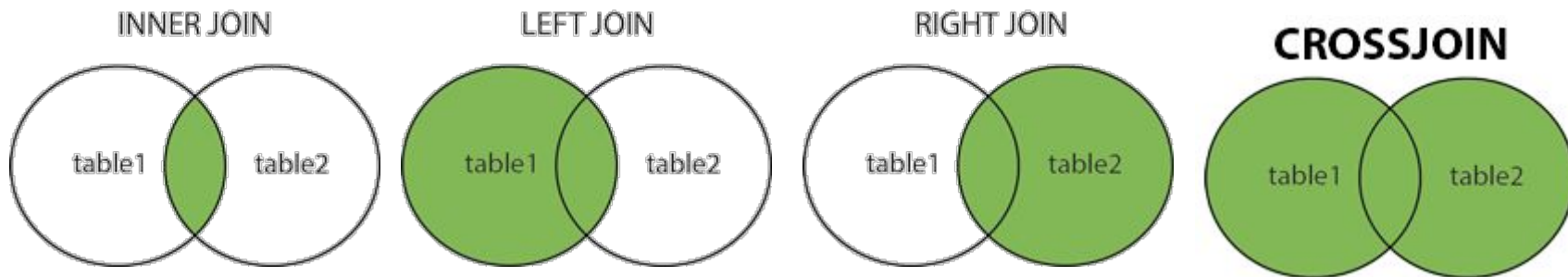
Fcpf	Nome_dependente	Sexo	Datanasc	Parentesco
33344555587	Alicia	F	05-04-1986	Filha
33344555587	Tiago	M	25-10-1983	Filho
33344555587	Janaina	F	03-05-1958	Esposa
98765432168	Antonio	M	28-02-1942	Marido
12345678966	Michael	M	04-01-1988	Filho
12345678966	Alicia	F	30-12-1988	Filha
12345678966	Elizabeth	F	05-05-1967	Esposa

---

# JOINS

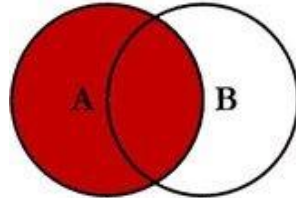
## SQL Join

A cláusula **JOIN** é usada para combinar linhas de duas ou mais tabelas, com base em uma coluna relacionada entre elas.

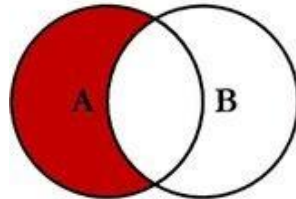


# SQL Join

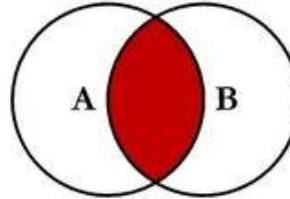
## SQL JOINS



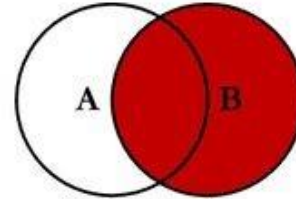
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



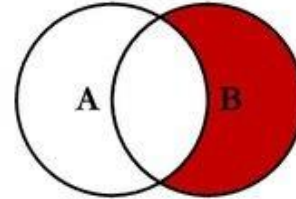
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL
```



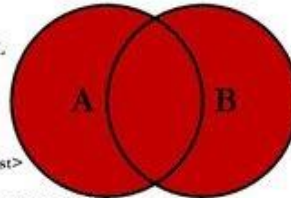
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



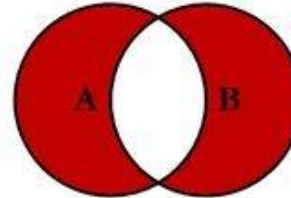
```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
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
```



## Tipos de Joins suportados no MySQL

**INNER JOIN:** Retorna registros que possuem valores correspondentes em ambas as tabelas

**LEFT JOIN:** Retorna todos os registros da tabela da esquerda e os registros correspondentes da tabela da direita

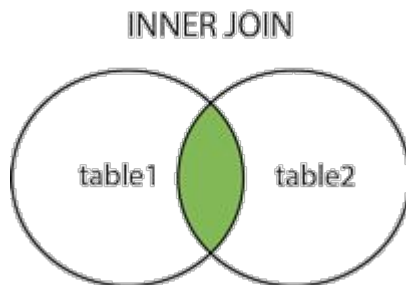
**RIGHT JOIN:** retorna todos os registros da tabela da direita e os registros correspondentes da tabela da esquerda

**CROSS JOIN:** Retorna todos os registros de ambas as tabelas

## SQL INNER JOIN

A cláusula **INNER JOIN** seleciona registros que têm valores correspondentes em ambas as tabelas.

Nota: A palavra-chave **INNER JOIN** seleciona todas as linhas de ambas as tabelas, desde que haja uma correspondência entre as colunas.





# SQL INNER JOIN

## INNER JOIN Syntax

**SELECT** column\_name(s)

**FROM** table1

**INNER JOIN** table2

**ON** table1.column\_name = table2.column\_name;

Selecionar o primeiro nome, último nome, endereço dos funcionários que trabalham no departamento de “Pesquisa”.





# SQL INNER JOIN

## INNER JOIN Syntax

**SELECT** column\_name(s)

**FROM** table1

**INNER JOIN** table2

**ON** table1.column\_name = table2.column\_name;

Liste o nome dos funcionários que estão desenvolvendo o “ProdutoX”.



# SQL INNER JOIN

## INNER JOIN Syntax

**SELECT** column\_name(s)

**FROM** table1

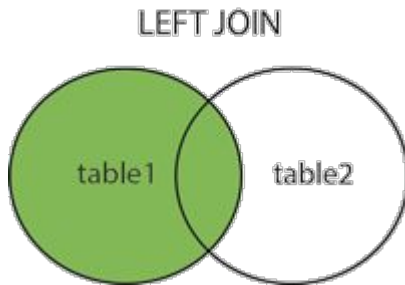
**INNER JOIN** table2

**ON** table1.column\_name = table2.column\_name;

Para cada projeto localizado em “Mauá”, liste o número do projeto, o número do departamento que o controla e o sobrenome, endereço e data de nascimento do gerente do departamento.

## SQL LEFT JOIN

A palavra-chave **LEFT JOIN** retorna todos os registros da tabela à esquerda (tabela1) e os registros correspondentes (se houver) da tabela à direita (tabela2).





# SQL LEFT JOIN

## LEFT JOIN Syntax

**SELECT** column\_name(s)

**FROM** table1

**LEFT JOIN** table2

**ON** table1.column\_name = table2.column\_name;

Liste o último nome de TODOS os funcionários e o último nome dos respectivos gerentes, caso possuam



# SQL LEFT JOIN

## LEFT JOIN Syntax

**SELECT** column\_name(s)

**FROM** table1

**LEFT JOIN** table2

**ON** table1.column\_name = table2.column\_name;

\*Encontre os departamentos que não possuem funcionários a eles vinculados

- Inserir 3 novos departamentos e 3 novos funcionários sem um departamento



# SQL LEFT JOIN

## LEFT JOIN Syntax

**SELECT** column\_name(s)

**FROM** table1

**LEFT JOIN** table2

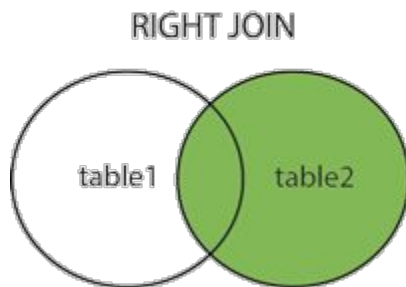
**ON** table1.column\_name = table2.column\_name;

\*Encontre os funcionários que não possuem um departamento a eles vinculado

- Inserir 3 novos departamentos e 3 novos funcionários sem um departamento

## SQL RIGHT JOIN

A palavra-chave **RIGHT JOIN** retorna todos os registros da tabela à direita (tabela2) e os registros correspondentes (se houver) da tabela à esquerda (tabela1).





# SQL RIGHT JOIN

RIGHT JOIN Syntax

**SELECT** column\_name(s)

**FROM** table1

**RIGHT JOIN** table2

**ON** table1.column\_name = table2.column\_name;

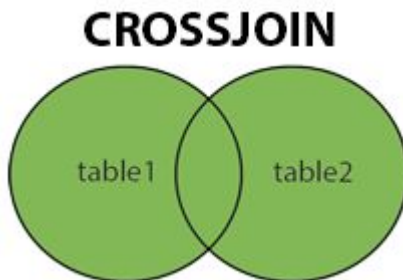
Encontre os departamentos que não possuem nenhum funcionário



## SQL CROSS JOIN

A palavra-chave CROSS JOIN retorna todos os registros de ambas as tabelas (tabela1 e tabela2).

Nota: CROSS JOIN pode potencialmente retornar conjuntos de resultados muito grandes!





# SQL CROSS JOIN

CROSS JOIN Syntax

**SELECT** column\_name(s)

**FROM** tabela1

**CROSS / FULL JOIN** tabela2

**CROS** tabela1.column\_name = tabela2.column\_name;

Teste entre as relações Funcionários e Departamento



## SQL Self JOIN

Uma junção automática é uma junção regular, mas a tabela é unida a si mesma.



# SQL Self JOIN

Self JOIN

**SELECT** column\_name(s)

**FROM** tabela1 **AS** T1

**JOIN** tabela1 **AS** T2

**ON** T1.column\_name = T2.column\_name;

Crie uma consulta que mostra apenas os funcionários que têm um supervisor.



## SQL UNION

O operador **UNION** é usado para combinar o conjunto de resultados de duas ou mais instruções **SELECT**.

Cada instrução **SELECT** dentro de **UNION** deve ter o mesmo número de colunas

As colunas também devem ter tipos de dados semelhantes

As colunas em cada instrução **SELECT** também devem estar na mesma ordem



# SQL UNION

## UNION Syntax

**SELECT** column\_name(s) FROM table1

**UNION**

**SELECT** column\_name(s) FROM table2;

Listar todos os nomes, sexo e data de nascimento de todas as pessoas do banco.



# SQL UNION

## UNION Syntax

**SELECT** column\_name(s) FROM table1

**UNION**

**SELECT** column\_name(s) FROM table2;

Imagine que a diretoria da empresa quer uma lista de todas as cidades onde a empresa possui alguma atividade, seja a localização de um departamento ou a localização de um projeto.



# SQL UNION

## UNION ALL

O operador **UNION** seleciona somente valores distintos. Para permitir valores duplicados use **UNION ALL**:





# SQL UNION

## UNION ALL Syntax

**SELECT** column\_name(s) FROM table1

## UNION ALL

**SELECT** column\_name(s) FROM table2;

Imagine que a diretoria da empresa quer uma lista de todas as cidades onde a empresa possui alguma atividade, seja a localização de um departamento ou a localização de um projeto.



# SQL EXCEPT

## EXCEPT Syntax

**SELECT** column\_name(s) FROM table1

**EXCEPT**

**SELECT** column\_name(s) FROM table2;

Listar os CPFs dos funcionários que não são gerentes de nenhum departamento.



# SQL EXCEPT

## EXCEPT Syntax

**SELECT** column\_name(s) FROM table1

**EXCEPT**

**SELECT** column\_name(s) FROM table2;

Encontre os Funcionários que NÃO são Supervisores



# SQL INTERSECT

**EXCEPT** Syntax

**SELECT** column\_name(s) FROM table1

**INTERSECT**

**SELECT** column\_name(s) FROM table2;

Encontre os Funcionários que são Supervisores



# SQL UNION/INTERSECT/EXCEPT

[VIDEO](#)

# SQL UNION/INTERSECT/EXCEPT

Os resultados das operações de multiconjunto da SQL. (a)

Duas tabelas, R(A) e S(A).

(b) R(A) UNION ALL S(A).

(c) R(A) EXCEPT ALL S(A).

(d) R(A) INTERSECT ALL S(A).

(a)

R	S
A	A
a1	a1
a2	a2
a2	a4
a3	a5

(b)

T
A
a1
a1
a2
a2
a3
a4
a5

(c)

T
A
a2
a3

(d)

T
A
a1
a2



## SQL Declaração GROUP BY

A instrução **GROUP BY** agrupa linhas com os mesmos valores em linhas de resumo, como "encontre o número de clientes em cada país".

A instrução **GROUP BY** é frequentemente usada com funções agregadas (**COUNT()**, **MAX()**, **MIN()**, **SUM()**, **AVG()**) para agrupar o conjunto de resultados por uma ou mais colunas.



# SQL Declaração GROUP BY

## GROUP BY Syntax

**SELECT** column\_name(s)

**FROM** table\_name

**WHERE** condition

**GROUP BY** column\_name(s)

**ORDER BY** column\_name(s);

Contar o número de funcionários por departamento





# SQL Declaração GROUP BY

## GROUP BY Syntax

**SELECT** column\_name(s)

**FROM** table\_name

**WHERE** condition

**GROUP BY** column\_name(s)

**ORDER BY** column\_name(s);

Somar os salários por departamento



# SQL Declaração GROUP BY

## GROUP BY Syntax

**SELECT** column\_name(s)

**FROM** table\_name

**WHERE** condition

**GROUP BY** column\_name(s)

**ORDER BY** column\_name(s);

Média de horas trabalhadas por projeto



# SQL Declaração GROUP BY

GROUP BY Syntax

**SELECT** column\_name(s)

**FROM** table\_name

**WHERE** condition

**GROUP BY** column\_name(s)

**ORDER BY** column\_name(s);

Quantidade de funcionários por sexo



# SQL Declaração GROUP BY

GROUP BY Syntax

**SELECT** column\_name(s)

**FROM** table\_name

**WHERE** condition

**GROUP BY** column\_name(s)

**ORDER BY** column\_name(s);

Maior salário em cada departamento



# SQL Declaração GROUP BY

GROUP BY Syntax

**SELECT** column\_name(s)

**FROM** table\_name

**WHERE** condition

**GROUP BY** column\_name(s)

**ORDER BY** column\_name(s);

Número de projetos em cada local



## SQL cláusula HAVING

A cláusula **HAVING** foi adicionada ao SQL porque a palavra-chave **WHERE** não pode ser usada com funções agregadas.



# SQL cláusula HAVING

HAVING Syntax

**SELECT** column\_name(s)

**FROM** table\_name

**WHERE** condition

**GROUP BY** column\_name(s)

**HAVING** condition

**ORDER BY** column\_name(s);

Encontrar departamentos com mais de 3 funcionários



# SQL cláusula HAVING

HAVING Syntax

**SELECT** column\_name(s)

**FROM** table\_name

**WHERE** condition

**GROUP BY** column\_name(s)

**HAVING** condition

**ORDER BY** column\_name(s);

Listar projetos que exigem mais de 100 horas de trabalho no total





## SQL Operador EXISTS

O operador **EXISTS** é usado para testar a existência de qualquer registro em uma subconsulta.

O operador **EXISTS** retorna **TRUE** se a subconsulta retornar um ou mais registros.



# SQL Operador EXISTS

## EXISTS Syntax

**SELECT** column\_name(s)

**FROM** table\_name

**WHERE EXISTS**

(**SELECT** column\_name **FROM** table\_name **WHERE** condition);

Listar funcionários que são gerentes de algum departamento



# SQL Operador EXISTS

## EXISTS Syntax

**SELECT** column\_name(s)

**FROM** table\_name

**WHERE EXISTS**

(**SELECT** column\_name **FROM** table\_name **WHERE** condition);

Listar departamentos que possuem projetos associados



## SQL Operadores ANY e ALL

Os operadores **ANY** e **ALL** permitem realizar uma comparação entre um único valor de coluna e um intervalo de outros valores.



## SQL Operador ANY

O operador **ANY**:

retorna um valor booleano como resultado

retorna **TRUE** se **QUALQUER** um dos valores da subconsulta atender à condição

**ANY** significa que a condição será verdadeira se a operação for verdadeira para qualquer um dos valores no intervalo.



# SQL Operador ANY

## ANY Syntax

**SELECT** column\_name(s)

**FROM** table\_name

**WHERE** column\_name operator **ANY**

(**SELECT** column\_name

**FROM** table\_name

**WHERE** condition);

Encontrar funcionários que ganham mais do que qualquer funcionário do departamento de 'Administração'



## SQL Operador ALL

O operador **ALL**:

retorna um valor booleano como resultado

retorna **TRUE** se **TODOS** os valores da subconsulta atenderem à condição

é usado com instruções **SELECT**, **WHERE** e **HAVING**

**ALL** significa que a condição será verdadeira somente se a operação for verdadeira para todos os valores no intervalo.



# SQL Operador ALL

ALL Syntax com **SELECT**

**SELECT ALL** column\_name(s)

**FROM** table\_name

**WHERE** condition;

Nota: O operador precisa ser um operação padrão de comparação.(=, <>, !=, >, >=, <, or <=).





# SQL Operador ALL

ALL Syntax com WHERE ou HAVING

**SELECT** column\_name(s)

**FROM** table\_name

**WHERE** column\_name operator **ALL**

(**SELECT** column\_name

**FROM** table\_name

**WHERE** condition);

Encontrar projetos que exigem mais horas do que todos os projetos no local 'São Paulo'



## Referência Bibliográfica

ELMASRI, Ramez; NAVATHE, Shamkant B.. Sistemas de banco de dados, 7ª ed., 2018

PUGA, Sandra; FRANÇA, Edson e GOYA, Milton. Banco de dados: Implementação em SQL, PL/SQL e Oracle 11g, 2013.

W3SCHOOL, MySQL Database, <https://www.w3schools.com/mysql/> acessado em 10/02/2023;