



Consultas Complexas

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

CONSULTAS



Consultas de recuperação básicas em SQL

A SQL tem uma instrução básica para recuperar informações de um banco de dados: a instrução **SELECT**. Existem muitas opções e tipos de instrução **SELECT** em SQL, nesta aula vamos utilizar os recursos da SQL para consultas de **recuperação simples**.



A estrutura **SELECT-FROM-WHERE** das consultas SQL básicas

A forma básica do comando **SELECT**, às vezes chamada de mapeamento ou bloco select-from-where, é composta pelas três cláusulas **SELECT**, **FROM** e **WHERE**, e tem a seguinte forma:

SELECT <Lista de atributos>

FROM <Lista de tabelas>

WHERE <Condição>;



A estrutura **SELECT-FROM-WHERE** das consultas SQL básicas

- <Lista de atributos> é uma lista de nomes de atributo cujos valores devem ser recuperados pela consulta.
- <Lista de Tabelas> é uma lista dos nomes de relação exigidos para processar a consulta.
- <Condição> é uma expressão condicional (booleana) que identifica as tuplas a serem recuperadas pela consulta.




***SQL Arithmetic Operators* - Operadores Aritméticos**

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide
%	Modulo



SQL *Bitwise Operators* - Operadores *Bitwise*

Operator	Description
&	Bitwise AND
	Bitwise OR
^	Bitwise exclusive OR




SQL *Comparison Operators* - Operadores de Comparação

Operator	Description
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to



SQL *Compound Operators* - Operadores Compostos

Operator	Description
<code>+=</code>	Add equals
<code>-=</code>	Subtract equals
<code>*=</code>	Multiply equals
<code>/=</code>	Divide equals
<code>%=</code>	Modulo equals
<code>&=</code>	Bitwise AND equals
<code>^-=</code>	Bitwise exclusive equals
<code> *=</code>	Bitwise OR equals



SQL *Logical Operators* - Operadores Lógicos

Operator	Description
ALL	TRUE if all of the subquery values meet the condition
AND	TRUE if all the conditions separated by AND is TRUE
ANY	TRUE if any of the subquery values meet the condition
BETWEEN	TRUE if the operand is within the range of comparisons
EXISTS	TRUE if the subquery returns one or more records
IN	TRUE if the operand is equal to one of a list of expression
LIKE	TRUE if the operand matches a pattern
NOT	Displays a record if the condition(s) is NOT TRUE
OR	TRUE if any of the conditions separated by OR is TRUE
SOME	TRUE if any of the subquery values meet the condition



SQL SELECT DISTINCT

A declaração **SELECT DISTINCT** é usada para retornar apenas valores distintos (diferentes). Dentro de uma tabela, uma coluna muitas vezes contém muitos valores duplicados e, às vezes, você só deseja listar os valores diferentes (distintos).



SQL SELECT DISTINCT

SELECT DISTINCT column1, column2, ...

FROM table_name;

Liste as diferentes faixas salariais dos funcionários.



SQL cláusula WHERE

A cláusula **WHERE** é usada para filtrar registros.

Ela é usada para extrair apenas os registros que atendem a uma condição especificada.

Observação: A cláusula WHERE não é usada apenas em declarações SELECT, ela também é usada em declarações UPDATE, DELETE, etc.!



SQL cláusula WHERE

SELECT column1, column2, ...

FROM table_name

WHERE condition;

Recupere todas as informações do(s) funcionais com primeiro nome “João”.



SQL AND, OR and NOT

A cláusula **WHERE** pode ser combinada com os operadores **AND**, **OR** e **NOT**.

Os operadores **AND** e **OR** são usados para filtrar registros com base em mais de uma condição:

- O operador **AND** exibe um registro se todas as condições separadas por **AND** forem **TRUE**.
- O operador **OR** exibe um registro se alguma das condições separadas por **OR** for **TRUE**.
- O operador **NOT** exibe um registro se a(s) condição(ões) for(em) **NOT TRUE**.



SQL AND, OR and NOT

AND Syntax

SELECT column1, column2, ...

FROM table_name

WHERE condition1 **AND** condition2 **AND** condition3 ...;

Liste os funcionários do sexo masculino com salário maior ou igual a 30.000,00R\$



SQL AND, OR and NOT

OR Syntax

SELECT column1, column2, ...

FROM table_name

WHERE condition1 **OR** condition2 **OR** condition3 ...;

Liste os funcionários que moram em São Paulo ou em Curitiba.



SQL AND, OR and NOT

NOT Syntax

SELECT column1, column2, ...

FROM table_name

WHERE NOT condition;

Liste os funcionários que não moram em São Paulo.



SQL ORDER BY

A palavra-chave **ORDER BY** é usada para classificar o conjunto de resultados em ordem ascendente ou descendente.

Por padrão, a palavra-chave **ORDER BY** classifica os registros em ordem ascendente. Para classificar os registros em ordem descendente, use a palavra-chave **DESC**.



SQL ORDER BY

SELECT column1, column2, ...

FROM table_name

ORDER BY column1, column2, ... **ASC|DESC**;

Preciso cortar orçamento, liste os funcionários em ordem decrescente de salário.



SQL Valores NULL

O que é um valor **NULL**?

Um campo com um valor **NULL** é um campo **sem valor**.

Se um campo em uma tabela for opcional, é possível inserir um novo registro ou atualizar um registro sem adicionar um valor a esse campo. Nesse caso, o campo será salvo com um valor **NULL**.

Observação: Um valor **NULL** é diferente de um valor zero ou de um campo que contém espaços. Um campo com um valor **NULL** é aquele que foi deixado em branco durante a criação do registro!



SQL Valores NULL

Como testar valores NULL?

Não é possível testar valores **NULL** com operadores de comparação, como `=`, `<` ou `<>`.

Em vez disso, devemos usar os operadores **IS NULL** e **IS NOT NULL**.



SQL Valores NULL

IS NULL Syntax

SELECT column_names

FROM table_name

WHERE column_name **IS NULL**;

Encontre os Funcionário que não possuem supervisor (Cpf_supervisor)



SQL Valores NULL

IS NOT NULL Syntax

SELECT column_names

FROM table_name

WHERE column_name **IS NOT NULL**;

Recupere o(s) nome(s) do(s) funcionário(s) que possuem supervisor.



SQL Cláusula **SELECT TOP**

A cláusula **SELECT TOP** é usada para especificar o número de registros a serem retornados.

A cláusula **SELECT TOP** é útil em tabelas grandes com milhares de registros. Retornar um grande número de registros pode afetar o desempenho.

Observação: Nem todos os sistemas de banco de dados suportam a cláusula **SELECT TOP**. O MySQL utiliza a cláusula **LIMIT** para selecionar um número limitado de registros, enquanto o Oracle utiliza **FETCH FIRST *n* ROWS ONLY** e **ROWNUM**.



SQL Cláusula SELECT TOP

MySQL Syntax:

SELECT column_name(s)

FROM table_name

WHERE condition

LIMIT number;

Recupere o registro dos 3 funcionários que têm o maior salário.



SQL Função MIN() MAX()

A função **MIN()** retorna o valor mais baixo da coluna selecionada.

A função **MAX()** retorna o valor mais alto da coluna selecionada.



SQL Função MIN() MAX()

MIN() Syntax

SELECT MIN(column_name)

FROM table_name

WHERE condition;

Recupere as informações do funcionário com o menor salário.



SQL Função MIN() MAX()

MAX() Syntax

```
SELECT MAX(column_name)
```

```
FROM table_name
```

```
WHERE condition;
```

Recupere as informações do funcionário com o maior salário.



SQL Funções **COUNT()**, **AVG()** e **SUM()**

A função **COUNT()** retorna o número de linhas que correspondem a um critério especificado.

A função **AVG()** retorna o valor médio de uma coluna numérica.

A função **SUM()** retorna a soma total de uma coluna numérica.



SQL Funções COUNT(), AVG() e SUM()

COUNT() Syntax

SELECT COUNT(column_name)

FROM table_name

WHERE condition;

Quantos funcionários possuímos cadastrados no banco?



SQL Funções COUNT(), AVG() e SUM()

AVG() Syntax

SELECT **AVG**(column_name)

FROM table_name

WHERE condition;

Qual é a média salarial dos meus funcionários?



SQL Funções COUNT(), AVG() e SUM()

SUM() Syntax

SELECT SUM(column_name)

FROM table_name

WHERE condition;

Qual é o meu custo mensal com folha de pagamento dos funcionários?



SQL Operador LIKE

O operador **LIKE** é usado em uma cláusula **WHERE** para pesquisar um padrão específico em uma coluna.

Existem dois curingas comumente usados em conjunto com o operador **LIKE**:

- O sinal de porcentagem (%) representa zero, um ou vários caracteres.
- O sinal de sublinhado (_) representa um único caractere.

OBS: Você também pode combinar qualquer número de condições usando os operadores **AND** ou **OR**.



SQL Operator LIKE

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"



SQL Operador LIKE

LIKE Syntax

SELECT column1, column2, ...

FROM table_name

WHERE column **LIKE** pattern;

Recupere o(s) funcionário(s) nascido(s) no ano de 72.



SQL Operador IN

O operador **IN** permite que você especifique vários valores em uma cláusula **WHERE**.

O operador **IN** é uma forma abreviada para múltiplas condições **OR**.



SQL Operador IN

IN Syntax

SELECT column_name(s)

FROM table_name

WHERE column_name **IN** (value1, value2, ...);

Recupere as informações dos funcionários que recebem 25000 e 30000 R\$



SQL Operador IN

IN Syntax

SELECT column_name(s)

FROM table_name

WHERE column_name **IN** (**SELECT** STATEMENT);

Recupere os registros dos funcionários que trabalham (TRABALHA_EM) no mesmo projeto e na mesma quantidade de horas do “Fernando” (Fcpf = “33344555587”)



SQL Operador BETWEEN

O operador **BETWEEN** seleciona valores **dentro de um determinado intervalo**. Os valores podem ser números, texto ou datas.

O operador **BETWEEN** é inclusivo: os valores de início e fim estão incluídos.



SQL Operador BETWEEN

BETWEEN Syntax

SELECT column_name(s)

FROM table_name

WHERE column_name **BETWEEN** value1 **AND** value2;

Recuperar todos os funcionários no departamento 5 cujo salário esteja entre R\$ 30.000 e R\$ 40.000



SQL Aliases

Aliases SQL são usados para dar um **nome temporário** a uma tabela ou a uma coluna em uma tabela.

Os aliases são frequentemente usados para tornar os nomes das colunas mais legíveis.

Um alias existe apenas durante a execução daquela consulta.

Um alias é criado com a palavra-chave **AS**.



SQL Aliases

Alias Column Syntax

SELECT column_name **AS** alias_name

FROM table_name;

Recupere Pnome como Nome e Unome como Sobrenome da tabela Funcionario.



SQL Aliases

Alias Table Syntax

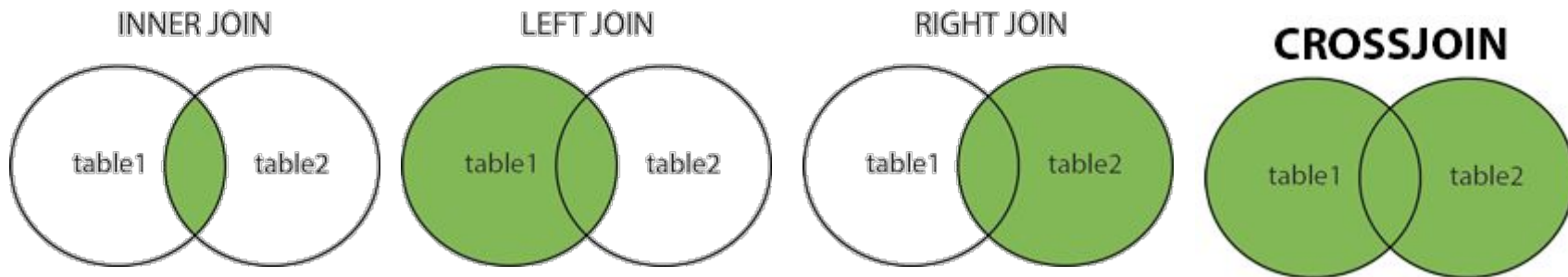
SELECT column_name

FROM table_name **AS** alias_name;

Recupere o nome de todos os funcionários que fazem parte do departamento de “Pesquisa”.

SQL Join

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





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

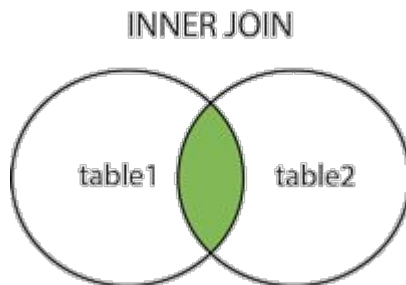
RIGHT JOIN: retorna todos os registros da tabela da direita e os registros correspondentes da tabela da esquerda

CROSS JOIN: Retorna todos os registros de ambas as tabelas

SQL INNER JOIN

The **INNER JOIN** keyword selects records that have matching values in both tables.

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

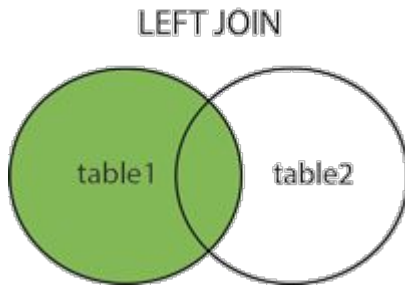
INNER JOIN table2

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

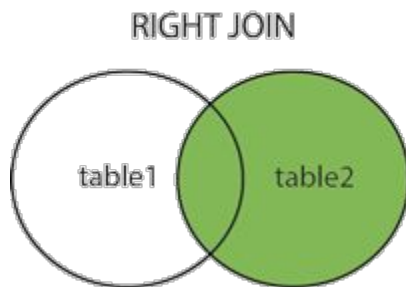
LEFT JOIN table2

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 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

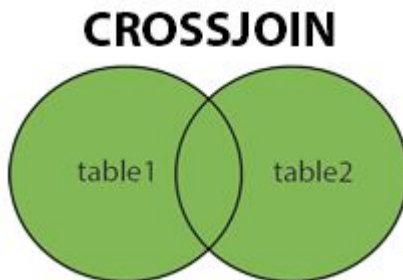
RIGHT JOIN table2

ON table1.column_name = table2.column_name;

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 table1
```

```
CROSS JOIN table2;
```

Teste entre as relações Funcionário e Dependentes



SQL Self JOIN

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



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;



SQL UNION

UNION ALL Syntax

The **UNION** operator selects only distinct values by default. To allow duplicate values, use **UNION ALL**:

```
SELECT column_name(s) FROM table1
```

```
UNION ALL
```

```
SELECT column_name(s) FROM table2;
```

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
a2
a3
a4
a5

(c)

T
A
a2
a3

(d)

T
A
a1
a2



SQL UNION/INTERSECT/EXCEPT

[VIDEO](#)



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);

Conte quantos funcionários moram em São Paulo, SP



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);

Liste as localidades com mais de 1 funcionário



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 Operator EXISTS

EXISTS Syntax

SELECT column_name(s)

FROM table_name

WHERE EXISTS

(**SELECT** column_name **FROM** table_name **WHERE** condition);



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 L

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 Operator ANY

ANY Syntax

SELECT column_name(s)

FROM table_name

WHERE column_name operator ANY

(**SELECT** column_name

FROM table_name

WHERE condition);



SQL Operador ANY L

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 ANY L

ALL Syntax com SELECT

SELECT ALL column_name(s)

FROM table_name

WHERE condition;

Note: The operator must be a standard comparison operator (=, <>, !=, >, >=, <, or <=).



SQL Operador ANY L

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);



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;