**Operadores lógicos**

O uso de um operador lógico faz com que duas condições tenham de produzir um resultado único. Uma linha só poderá ser retornada se o resultado global da condição for verdadeiro.

A tabela abaixo mostra os operadores lógicos disponíveis em SQL:

* AND Retorna TRUE se ambas as condições forem verdadeiras
* OR Retorna TRUE se uma das condições for verdadeira
* NOT Retorna TRUE se a condição seguinte for falsa

No exemplo abaixo uma linha será retornada se o empregado for do estado de SP e o seu salário maior que 2200.

1. SELECT nome, salário, cidade, estado
2. FROM empregados
3. WHERE estado=\*\*\*SP\*\*\* and salario > 2200

Para produzir um resultado diferente basta alterar esta consulta, trocando o operador AND por OR.

1. SELECT nome, salário, cidade, estado
2. FROM empregados
3. WHERE estado=\*\*\*SP\*\*\* or salario > 2200

Neste caso uma linha será retornado se o empregado for do estado de SP ou se o seu salário for maior que 2200.

**Leitura Complementar - LIKE**

Cada *padrão* define um conjunto de cadeias de caracteres. A expressão LIKE retorna verdade se a *cadeia\_de\_caracteres* estiver contida no conjunto de cadeias de caracteres representado pelo *padrão*; como esperado, a expressão NOT LIKE retorna falso quando LIKE retorna verdade, e vice-versa, e a expressão equivalente é NOT (*cadeia\_de\_caracteres* LIKE *padrão*).

Quando o *padrão* não contém os caracteres percentagem ou sublinhado, o *padrão* representa apenas a própria cadeia de caracteres; neste caso LIKE atua como o operador igual. No *padrão* o caractere sublinhado (\_) representa (corresponde a) qualquer um único caractere; o caractere percentagem (%) corresponde a qualquer cadeia com zero ou mais caracteres.

Alguns exemplos:

1. 'abc' LIKE 'abc' verdade
2. 'abc' LIKE 'a%' verdade
3. 'abc' LIKE '\_b\_' verdade
4. 'abc' LIKE 'c' falso

A correspondência com padrão LIKE sempre abrange toda a cadeia de caracteres. Para haver correspondência com o padrão em qualquer posição da cadeia de caracteres, o padrão deve começar e terminar pelo caractere percentagem.

Para corresponder ao próprio caractere sublinhado ou percentagem, sem corresponder a outros caracteres, estes caracteres devem ser precedidos pelo caractere de escape no *padrão*. O caractere de escape padrão é a contrabarra, mas pode ser definido um outro caractere através da cláusula ESCAPE. Para corresponder ao próprio caractere de escape, devem ser escritos dois caracteres de escape.

Deve ser observado que a contrabarra também possui significado especial nos literais cadeias de caracteres e, portanto, para escrever em uma constante um padrão contendo uma contrabarra devem ser escritas duas contrabarras no comando SQL. Assim sendo, para escrever um padrão que corresponda ao literal contrabarra é necessário escrever quatro contrabarras no comando, o que pode ser evitado escolhendo um caractere de escape diferente na cláusula ESCAPE; assim a contrabarra deixa de ser um caractere especial para o LIKE (Mas continua sendo especial para o analisador de literais cadeias de caracteres e, por isso, continuam sendo necessárias duas contrabarras).

REGEXP –

todo primeiro nome q comece com a RGEXP ‘^a’

todo primeiro nome q comece com a e d - RGEXP ‘^a|^d’

todo primeiro nome q comece com ga e ge - RGEXP ‘^[ger]a’

**Operadores Matematicos**

**Usando operadores aritméticos.**

Para criar expressões aritméticas em uma consulta SQL usamos os operadores abaixo:

* + (somar)
* - (subtrair)
* \* (multiplicar)
* / (dividir)

Os operadores acima podem ser usados apenas em colunas do tipo numérico.

1. SELECT nome, salario, salario\*12 As salario\_anual
2. FROM empregados

**Operadores de comparação**

**Operadores de comparação**

Os operadores de comparação são usados em condições que comparam uma expressão a outro valor ou expressão. A tabela abaixo mostra os operadores:

* = Igual a
* > Maior que
* >= Maior ou igual a que
* < Menor que
* <= Menor ou igual a que
* <> Diferente de

1. SELECT codigo\_empregado, nome, salario
2. FROM empregados
3. WHERE codigo\_empregado=12

No exemplo acima, usamos o igual para criar um filtro na cláusula where, este filtro diz que os dados retornados deverão ser do código de empregado IGUAL a 12.

**Operadores lógicos**

**Operadores lógicos**

O uso de um operador lógico faz com que duas condições tenham de produzir um resultado único. Uma linha só poderá ser retornada se o resultado global da condição for verdadeiro.

A tabela abaixo mostra os operadores lógicos disponíveis em SQL:

* AND Retorna TRUE se ambas as condições forem verdadeiras
* OR Retorna TRUE se uma das condições for verdadeira
* NOT Retorna TRUE se a condição seguinte for falsa

No exemplo abaixo uma linha será retornada se o empregado for do estado de SP e o seu salário maior que 2200.

1. SELECT nome, salário, cidade, estado
2. FROM empregados
3. WHERE estado=\*\*\*SP\*\*\* and salario > 2200

Para produzir um resultado diferente basta alterar esta consulta, trocando o operador AND por OR.

1. SELECT nome, salário, cidade, estado
2. FROM empregados
3. WHERE estado=\*\*\*SP\*\*\* or salario > 2200

Neste caso uma linha será retornado se o empregado for do estado de SP ou se o seu salário for maior que 2200.

**Alias – apelido da tabela**

Customer c

Payment p

Join – une 2 tabelas JOIN ..... ON / JOIN – une multiplas tabelas

LEFT /RIGHT JOIN

UNION – une dois selects em um só

**Leitura Adicional: Union**

O operador UNION combina os resultados de duas ou mais queries em um único result set, retornando todas as linhas pertencentes a todas as queries envolvidas na execução. Para utilizar o UNION, o número e a ordem das colunas precisam ser idênticos em todas as queries e os data types precisam ser compatíveis.

O operador UNION, por default, executa o equivalente a um SELECT DISTINCT no result set final. Em outras palavras, ele combina o resultado de execução das duas queries e então executa um SELECT DISTINCT a fim de eliminar as linhas duplicadas. Este processo é executado mesmo que não hajam registros duplicados.

USE sakila;

SELECT \* FROM sakila.language;

INSERT INTO language

VALUES(Default, 'Portuguese', '2008-02-10 05:02:19');

INSERT INTO language

VALUES

(Default, 'Spanish33', '2009-02-10 05:02:19'),

(Default, 'Polish33', '2010-02-10 05:02:19')

;

USE sakila;

SELECT \* FROM sakila.city;

SELECT \* FROM sakila.country;

INSERT INTO country

VALUES

(Default, 'Brasil2', '2035-02-15 04:45:25')

;

INSERT INTO city

VALUES

(Default, 'Sao Paulo2', last\_insert\_id(), '2034-02-15 04:45:25')

;

Criando backup de uma tabela –

CREATE TABLE payment\_backup AS

SELECT \* FROM sakila.payment;

Atualizar a tabela

UPDATE payment

SET

amount = 15.99

WHERE

payment\_id = 1

DELETE FROM payment

WHERE payment\_id = 16049

Funções

<https://www.w3schools.com/sql/sql_ref_sqlserver.asp>

SELECT

MAX(amount) AS Maior,

MIN(amount) AS Menor,

AVG(amount) AS 'Media de Valores'

FROM payment

SELECT

MAX(amount) AS Maior,

MIN(amount) AS Menor,

AVG(amount) AS 'Media de Valores',

COUNT(amount) AS 'NUmeros de vendas'

FROM payment

SELECT

cus.customer\_id,

cus.first\_name,

cus.last\_name,

SUM(amount) AS total

FROM payment pay

JOIN customer cus USING(customer\_id)

GROUP BY customer\_id

ORDER BY total DESC

SELECT

cus.customer\_id,

cus.first\_name,

cus.last\_name,

SUM(amount) AS total

FROM payment pay

JOIN customer cus USING(customer\_id)

GROUP BY customer\_id

HAVING total >= 150

SELECT

cus.customer\_id,

cus.first\_name,

cus.last\_name,

SUM(amount) AS total

FROM payment pay

JOIN customer cus USING(customer\_id)

GROUP BY customer\_id

HAVING total >= 150

ORDER BY total DESC

SELECT \*

FROM customer

WHERE customer\_id IN (

SELECT

customer\_id

FROM payment

GROUP BY customer\_id

HAVING COUNT(\*) > 35

)

SELECT \*

FROM customer

WHERE customer\_id = ANY (

SELECT

customer\_id

FROM payment

GROUP BY customer\_id

HAVING COUNT(\*) > 35

)

VIEW---

CREATE VIEW vendas\_por\_cliente AS

SELECT

cus.customer\_id,

cus.first\_name,

cus.last\_name,

pay.amount

FROM customer cus

JOIN payment pay

ON cus.customer\_id = pay.payment\_id

SELECT \* FROM sakila.vendas\_por\_cliente

WHERE amount >= 9.99

;

CREATE OR REPLACE VIEW vendas\_por\_cliente AS

SELECT

cus.customer\_id,

cus.first\_name,

cus.last\_name,

pay.amount

FROM customer cus

JOIN payment pay

ON cus.customer\_id = pay.payment\_id

ORDER BY pay.amount DESC;

Funções com Strings

<https://dev.mysql.com/doc/refman/8.0/en/string-functions.html>

TRIM remove espacos do inicio

BOTH remove espacos do inicio e final

LEADING remove espacos do inicio

TRAILING remove espacos do final

LOCATE() localiza onde a string esta = SELECT LOCATE(‘C’, ‘Carros’)

LOWERCASE = LCASE( ‘Carros’)

UPERCASE = SELECT UCASE( ‘Carros’)

DATA BASE

https://www.w3schools.com/sql/sql\_datatypes.asp

CREATE DATABASE carros;

USE carros;

CREATE TABLE marcas(

id int NOT NULL AUTO\_INCREMENT,

nome\_marca VARCHAR(255) NOT NULL,

PRIMARY KEY (id)

);

ALTER TABLE marcas

ADD origem VARCHAR(255);

CREATE TABLE inventario(

id int NOT NULL AUTO\_INCREMENT,

modelo VARCHAR(255) NOT NULL,

transmissao VARCHAR(255) NOT NULL,

motor VARCHAR(255) NOT NULL,

combustivel VARCHAR(255) NOT NULL,

marcas\_id int NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (marcas\_id) REFERENCES marcas(id)

);

CREATE TABLE clientes(

id int NOT NULL AUTO\_INCREMENT,

nome VARCHAR(255) NOT NULL,

sobrenome VARCHAR(255) NOT NULL,

endereco VARCHAR(255) NOT NULL,

PRIMARY KEY (id)

);

-------------------------------------------------------------------------------------------------------

INSERT INTO clientes(nome, sobrenome, endereco)

VALUES('Pollyana', 'Ferreira', 'Rua do Conde');

INSERT INTO clientes(nome, sobrenome, endereco)

VALUES('Eduardo', 'Ferreira', 'Rua do Morango');

INSERT INTO clientes(nome, sobrenome, endereco)

VALUES('Thyago', 'Ferreira', 'Rua do Marmelo');

INSERT INTO clientes(nome, sobrenome, endereco)

VALUES('Maria Eduarda', 'Martins', 'Rua Marco Aurelio');

INSERT INTO clientes(nome, sobrenome, endereco)

VALUES('Regina', 'Santana', 'Rua Castro Alves');

SELECT \* FROM carros.marcas;

-------------------------------------------------------------------------------------------------------

INSERT INTO marcas(nome\_marca, origem)

VALUES('Renault', 'Franca');

INSERT INTO marcas(nome\_marca, origem)

VALUES('Wolkswagem', 'Alemanha');

INSERT INTO marcas(nome\_marca, origem)

VALUES('Toyota', 'Japao');

INSERT INTO marcas(nome\_marca, origem)

VALUES('Hyundai', 'Coreia do Sul');

INSERT INTO marcas(nome\_marca, origem)

VALUES('Ford', 'Estados Unidos');

SELECT \* FROM carros.inventario;

-------------------------------------------------------------------------------------------------------

INSERT INTO inventario(modelo, transmissao, motor, combustivel, marcas\_id)

VALUES

('Duster', 'Automatica', '1.6', 'Flex', 1),

('Gol', 'Manual', '1.0', 'Flex', 2),

('Ethios', 'Automatico', '1.4', 'Flex', 3),

('HB20', 'Automatico', '1.8', 'Flex', 4),

('Ranger', 'Automatica', '3.0', 'Diesel', 5)

-------------------------------------------------------------------------------------------------------

USUÁRIOS

USE mysql;

SELECT \* FROM  user;

--ou

SELECT \* FROM  mysql.user;

CREATE USER ana IDENTIFIED BY '1212';

CREATE USER joao@localhost IDENTIFIED BY '1212';

CREATE USER priscila@andreia.cono.com IDENTIFIED BY '1212';

SELECT \* FROM  mysql.user;

-------------------------------------------------

DROP USER [priscila@andreia.cono.com](mailto:priscila@andreia.cono.com);

------------------------------------

SET PASSWORD FOR joao@localhost = '4444';

----------------------------------------

Adicionar privilegios

GRANT SELECT, INSERT, UPDATE, DELETE

ON sakila.\*

TO ana;

SHOW GRANTS FOR ana;

----------------------------------------

GRANT ALL

ON \*.\*

TO joao@localhost;

SHOW GRANTS FOR joao@localhost;

--------------------------------------------------------------

REVOKE

CREATE USER priscila IDENTIFIED BY '1212';

GRANT SELECT, INSERT, UPDATE

ON sakila.\*

TO priscila;

REVOKE UPDATE

ON sakila.\*

FROM priscila;

SHOW GRANTS FOR priscila;

SELECT \* FROM  mysql.user;