

Professor Maurício Buess

mbuess@up.edu.br

github.com/mauriciobuess



Objetivo:

- Compreender os operadores lógicos;
- Uso de operadores (matemáticos, relacionais e lógicos);
- Principais funções de agregação
- Clásula "group by"

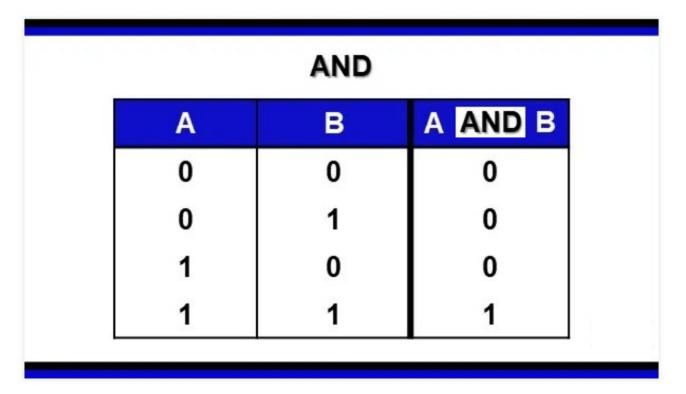


Introdução aos Operadores Lógicos

- Operadores lógicos são dispositivos usados nas operações entre dados lógicos;
- Dados lógicos são aqueles dados cujo valores são dicotômicos;
- Consequentemente, expressões lógicas são expressões compostas por dados e operadores lógicos cujo resultado é um dado dicotômico;
- Os três principais operadores lógicos são:
 - AND
 - OR
 - NOT



Operador Lógico AND



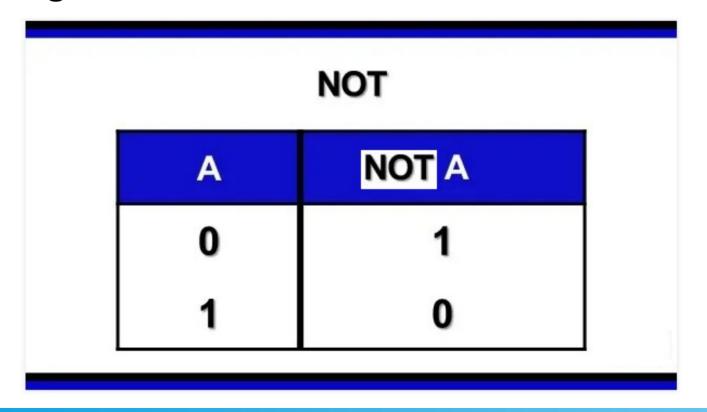


Operador Lógico OR

OR A B A OR B 0 0 0 0 1 1 1 0 1



Operador Lógico NOT





Exemplos

- -- Encontrar todos os empregados que têm salário maior que 50000 e que trabalham no
- -- departamento 3

SELECT * FROM empregados

WHERE salario > 50000 AND departamento_id = 3;

- -- Encontrar todos os empregados que trabalham no departamento 3 ou que têm
- -- um salário maior que 50000

SELECT * FROM empregados

WHERE departamento_id = 3 OR salario > 50000;

-- Encontrar todos os empregados que não trabalham no departamento 3

SELECT * FROM empregados

WHERE NOT departamento_id = 3;



Exercício:

- 1) Crie um banco de dados de nome "Aula07"; create database Aula07;
- 2) Crie a tabela cliente com a seguinte estrutura:
 - IdCliente int auto incremento e chave primária
 - cliente varchar(100) não nulo
 - uf char(02) não nulo
 - dataNascimento date nulo

```
use Aula07;
create table cliente (idCliente int primary key auto_increment
,cliente varchar(100) not null ,uf char(02) not null ,dataNascimento date
,idade int);
```



Exercício:

```
3) Insira alguns dados.
insert into cliente (cliente, uf, dataNascimento, idade)
values ('Ana Flavia', 'SP', '1998-05-14',26)
, ('Romeu Cunha', 'PR', '2007-03-03', 47)
, ('Moises Aguasanta', 'SP', '2001-03-30', 23)
, ('Leonardo Caprio', 'RJ', '2004-09-18', 19)
, ('Janete Caldeirao', 'SP', '1995-02-10', 29);
```

4) Escreva uma consulta para encontrar todos os clientes com idade entre 20 e 30 anos.

```
select *
from cliente
where idade >= 20
and idade <= 30:
```



Operadores Matemáticos e Relacionais na Cláusula WHERE

- Operadores Matemáticos: +, -, *, /
- Operadores Relacionais: =, !=, <, >, <=, >=

Exemplos:

```
select *
from cliente
where idade >= 20
and idade < 29;

select *
from cliente
where (idade + 1) < 21;
```



Exercício:

- 1) Crie uma tabela chamada produto com a seguinte estrutura:
 - IdProduto int chave primária auto incremento
 - produto varchar(100) não nulo
 - qtdEstoque int n\u00e3o nulo
 - vlrVenda decimal(10,02)
- 2) Inclua sete produtos quaisquer.
- 3) Escreva uma consulta que retorne os produtos que tenham estoque cuja valor de venda acrescido em 10% seja menor que \$ 100,00.
- 4) Atualize o preço desses produtos para \$99.99.



Exercício:

- 1) Crie uma tabela chamada produto com a seguinte estrutura:
 - idProduto int chave primária auto incremento
 - produto varchar(100) não nulo
 - qtdEstoque int n\u00e3o nulo
 - vlrVenda decimal(10,02)

```
create table produto (idProduto int primary key auto_increment
,produto varchar(100) not null
,qtdEstoque int not null
,vlrVenda decimal(10,02));
```



Exercício:

2) Inclua sete produtos quaisquer.

```
insert into produto (produto ,qtdEstoque, vlrVenda) values ('Rosquinha Kero Mais', 300, 15.50) , ('Parafuso torto', 1500, 99.50) , ('Coelhinho da Pascoa', 33, 2499.99) , ('Atestado médico (1 dia)', 10, 149.99) , ('Bola quadrada', 250, 269.70) , ('Porca grande (viva)', 2, 1500) , ('Porca pequena (viva)', 0, 500);
```



Exercício:

3) Escreva uma consulta que retorne os produtos que tenham estoque cuja valor de venda acrescido em 10% seja menor que \$ 100,00.

```
select *
from produto
where qtdEstoque > 0
and (vlrVenda*1.1) < 100
```



Exercício:

4) Atualize o preço desses produtos para \$99.99.

```
update produto
set vlrVenda = 99.99
where qtdEstoque > 0
and (vlrVenda*1.1) < 100;
```



Funções de agregação:

- Entende-se por função um recurso que retorna um resultado a ser processado;
- O resultado retornado por uma função é de um tipo de dado válido ao ambiente em que a mesma está inserida (vide aula de tipo de dados);
- Funções de agregação são funções utilizadas em SQL para realizar cálculos e resumir dados em grupos ou conjuntos de registros. Elas operam em um conjunto de valores e retornam um único valor que representa um resumo ou estatística do conjunto.



Principais Funções de agregação:

- COUNT(): Conta o número de linhas.
- SUM(): Soma valores numéricos.
- AVG(): Calcula a média dos valores numéricos.
- MIN(): Encontra o valor mínimo.
- MAX(): Encontra o valor máximo
- GROUP_CONCAT(): Concatena valores de uma coluna em uma única string, separados por um delimitador. É útil para criar listas ou reunir valores em uma única célula.



Principais Funções de agregação:

- COUNT(): Conta o número de linhas.
 SELECT COUNT(*) FROM tabela;
 SELECT COUNT(coluna) FROM tabela;
- SUM(): Soma os valores de uma coluna numérica. É útil para calcular totais. SELECT SUM(coluna) FROM tabela;
- AVG(): Calcula a média dos valores numéricos.
 SELECT AVG(coluna) FROM tabela;



Principais Funções de agregação:

- MIN(): Retorna o menor valor em uma coluna.
 SELECT MIN(coluna) FROM tabela;
- MAX(): Retorna o maior valor em uma coluna.
 SELECT MAX(coluna) FROM tabela;
- GROUP_CONCAT(): Concatena valores de uma coluna em uma única string, separados por um delimitador. É útil para criar listas ou reunir valores em uma única célula.

SELECT GROUP_CONCAT(coluna SEPARATOR ', ') FROM tabela;



Exercícios:

- 1) Conte quantos produtos estão sem estoque;
- 2) Calcule o valor total imobilizado pelo estoque;
- 3) Qual o valor médio de venda praticado pela empresa?
- 4) Qual é o produto mais barato e o mais caro?



Exercícios:

- Conte quantos produtos estão sem estoque;
 select COUNT(*) from produto where qtdEstoque <= 0;
 select * from produto where qtdEstoque <= 0;
- 2) Calcule o valor total imobilizado pelo estoque considerando o valor de venda; select SUM(qtdEstoque * vlrVenda) from produto;
- 3) Qual o valor médio de venda praticado pela empresa? select AVG(vlrVenda) from produto;
- 4) Qual é o produto mais barato e o mais caro?
 select min(vlrVenda) 'mais barato'
 , max(vlrVenda) 'mais caro'
 from produto;



Importante

- As funções de agregação são essenciais para realizar análises e gerar relatórios a partir dos dados armazenados em um banco de dados.
- Elas ajudam a transformar grandes conjuntos de dados em informações compreensíveis e úteis.



Cláusula GROUP BY

- Usada para agrupar linhas que têm os mesmos valores em colunas especificadas em um conjunto de resultados.
- Muito utilizada em combinação com funções de agregação para calcular valores resumidos para cada grupo de dados.
- A cláusula GROUP BY organiza os resultados da consulta em grupos com base nos valores de uma ou mais colunas. Após a aplicação do GROUP BY, pode-se usar funções de agregação como COUNT(), SUM(), AVG(), MIN(), e MAX() para calcular estatísticas para cada grupo.



Cláusula GROUP BY

SELECT coluna1, coluna2, função_agregação(coluna3) FROM tabela GROUP BY coluna1, coluna2;

- coluna1, coluna2: As colunas usadas para agrupar os dados. Todos os dados com os mesmos valores nessas colunas serão agrupados.
- função_agregação(coluna3): A função de agregação que realiza cálculos sobre os dados agrupados.

tabela: O nome da tabela de onde os dados são selecionados.



Cláusula GROUP BY

Exemplo:

 Totalizar os clientes da empresa por UF (unidade federativa): select uf, count(1) 'Tot.Clientes' from cliente group by uf;

 Totalizar os clientes da empresa por UF destacando a média de idade: select uf, count(1) 'Tot.Clientes'
 , AVG(idade) 'media idade'
 from cliente
 group by uf



Cláusulas GROUP BY e ORDER BY

 Totalizar os clientes da empresa por UF (unidade federativa): select uf, count(1) 'Tot.Clientes' from cliente group by uf order by 2 ASC;

 Totalizar os clientes da empresa por UF destacando a média de idade: select uf, count(1) 'Tot.Clientes'
 , AVG(idade) 'media idade'
 from cliente
 group by uf

order by 3 DESC;



Importante:

- Colunas na SELECT: Todas as colunas na cláusula SELECT que não são usadas em funções de agregação devem estar na cláusula GROUP BY. Caso contrário, o MySQL retornará um erro.
- Funções de Agregação: Sem GROUP BY, funções de agregação como SUM() e AVG() calculam resultados para todo o conjunto de dados. Com GROUP BY, elas calculam resultados para cada grupo individualmente.

A cláusula GROUP BY é uma ferramenta poderosa para análise e sumarização de dados em SQL, permitindo gerar relatórios e insights a partir de grandes conjuntos de informações.