

11 – Introdução

- Muitos desenvolvedores acreditam que é possível utilizar operadores condicionais (IF, CASE) apenas dentro de sub rotinas do banco de dados, tais como triggers e stored procedures.
- Entretanto, estes operadores condicionais podem ser utilizados em comandos SELECT, aumentando seu poder de retornar dados.
- Estes operadores condicionais também são conhecidos como Flow Control Operators (Operadores de controle de Fluxo), sendo formados basicamente por 4 comandos:

1. **IF()**

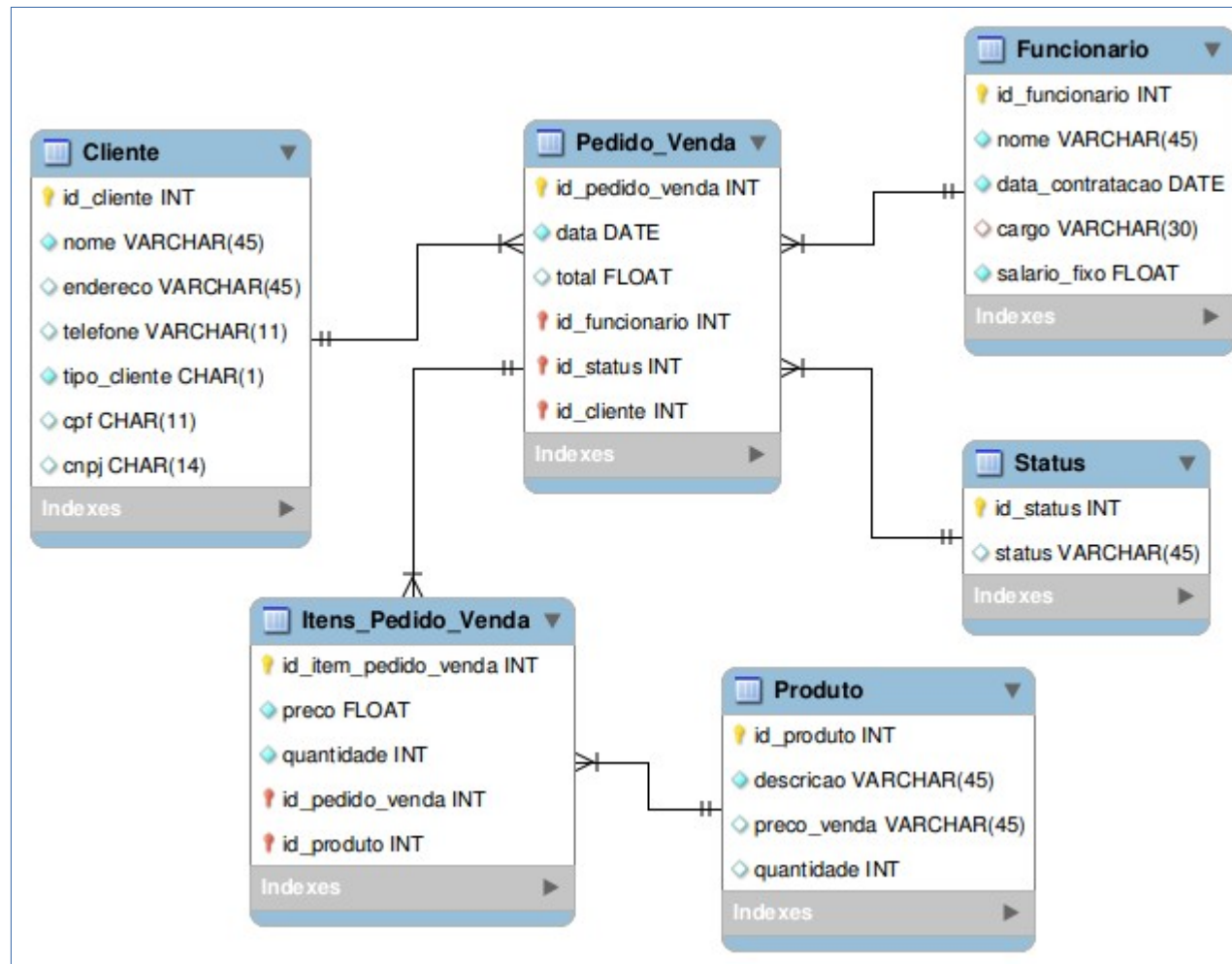
2. **CASE**

3. **IFNULL()**

4. **NULLIF()**

11 – Introdução

- Para exemplificar consultas com operadores condicionais, vamos utilizar o banco de dados mostrado na figura abaixo:



11.1 – Operador IF

- O operador IF pode ser usado para inserir condições “SE” no SELECT, aumentando as possibilidades de deixar os resultados mais dinâmicos. A sintaxe do IF é mostrado no quadro abaixo:

SELECT IF(A, B, C) FROM tabela

//Se A for verdadeiro, o resultado será B, senão o resultado será C

Ex: **SELECT IF(1<2, Sim, Não) FROM tabela**

Resultado=Sim

Ex: **SELECT IF(5<3, Sim, Não) FROM tabela**

Resultado=Não

11.1 – Operador IF

- O exemplo abaixo mostra uma consulta onde os funcionários que possuem o cargo de gerente possuem seu salário somado em 500 reais. Para os demais funcionários é exibido apenas o salário.

```
SELECT nome, cargo, IF(cargo='Gerente', salario_fixo+500, salario_fixo) AS 'Salário' FROM  
funcionario;
```

#	nome	cargo	Salário
1	Paulo Medeiros	Vendedor	800
2	Maria da Silva	Zeladora	1000
3	Gabriel Jesus	Vendedor	850
4	Zé Roberto	Gerente	2500
5	Matheus Sales	Vendedor	900

11.1 – Operador IF

- O exemplo abaixo mostra uma consulta onde os produtos que possuem quantidade inferior a 30 será atribuído o valor 'Estoque Baixo' para o campo virtual 'Situação'. Do contrário será exibido o valor 'Estoque Alto'.

```
SELECT id_produto, descricao, quantidade, IF(quantidade<=30, 'Estoque Baixo', 'Estoque Alto')  
AS 'Situação' FROM produto;
```

id_produto	descricao	quantidade	Situação
1	Mouse	50	Estoque Alto
2	Teclado	15	Estoque Baixo
3	Monitor...	20	Estoque Baixo
4	Monitor...	5	Estoque Baixo
5	Notebo...	10	Estoque Baixo
6	Notebo...	60	Estoque Alto

11.1 – Operador IF

- O exemplo abaixo possui um diferencial com relação ao anterior, mostrando o valor 'Estoque Médio' para o campo virtual 'Situação' caso a quantidade varie entre 31 e 50. Neste caso, é usado um IF dentro de outro IF.

```
SELECT id_produto, descricao, quantidade, IF(quantidade<=30, 'Estoque Baixo',  
IF(quantidade>30 AND quantidade<=50, 'Estoque Médio', 'Estoque Alto')) AS 'Situação' FROM  
produto;
```

id_produto	descricao	quantidade	Situação
1	Mouse	50	Estoque Médio
2	Teclado	15	Estoque Baixo
3	Monitor...	20	Estoque Baixo
4	Monitor...	5	Estoque Baixo
5	Notebo...	10	Estoque Baixo
6	Notebo...	60	Estoque Alto

11.1 – Operador IF

- O exemplo abaixo mostra uma consulta onde se verifica o total que cada vendedor vendeu na tabela de pedidos, caso ele tenha vendido mais de 500, seus status será bom, senão ruim.

```
SELECT f.id_funcionario, nome, IF ((SELECT SUM(total) FROM pedido_venda p WHERE  
p.id_funcionario=f.id_funcionario)>500,'Bom','Ruim') AS 'Status' FROM funcionario f;
```

id_funcionario	nome	Status
1	Paulo Medeiros	Bom
2	Maria da Silva	Ruim
3	Gabriel Jesus	Ruim
4	Zé Roberto	Ruim
5	Matheus Sales	Ruim

11.1 – Operador IF

- O exemplo abaixo faz a totalização da quantidade de pedidos existente para cada um dos status possíveis. No caso, pedidos com id_status=1 estão em aberto e id_status=2 estão fechados.

```
SELECT  
  
    SUM(IF(id_status =1, 1, 0)) AS 'Aberto',  
    SUM(IF(id_status =2, 1, 0)) AS 'Fechado'  
  
FROM  
  
    pedido_venda;
```

Aberto	Fechado
1	3

11.1 – Operador IF

- Uma outra forma de fazer a totalização da quantidade de pedidos existente para cada um dos status possíveis é utilizar o comando COUNT, como mostra o exemplo abaixo:

```
SELECT  
COUNT(IF(id_status=1, 1, NULL)) AS 'Aberto',  
COUNT(IF(id_status=2, 1, NULL)) AS 'Fechado'  
FROM  
pedido_venda;
```

Aberto	Fechado
1	3

11.2 – Operador CASE

- Enquanto o operador IF é utilizado para verificar apenas uma condição com poucos elementos, o operador **CASE** é utilizado para condições que possuam vários elementos condicionais.

SELECT

CASE

WHEN <campo> = 'A' **THEN** 'X'

WHEN <campo> = 'B' **THEN** 'Y'

ELSE 'Z'

END

FROM <tabela>;

//Se <campo> for igual A, o resultado será X

//Se <campo> for igual B, o resultado será Y

//Se <campo> não for igual A ou B, o resultado será Z

11.2 – Operador CASE

- O exemplo abaixo mostra a descrição do status dos pedidos. Caso o id_status=1 o resultado será 'Aberto', para id_status=2 o resultado será 'Fechado', e se não for 1 ou 2 o resultados será 'Indefinido':

```
SELECT id_pedido_venda, data, id_cliente, total,  
CASE  
  WHEN id_status=1 THEN 'Aberto'  
  WHEN id_status=2 THEN 'Fechado'  
  ELSE 'Indefinido'  
END AS 'Status'  
FROM pedido_venda;
```

id_pedido_venda	data	id_cliente	total	id_status	Status
1	2016-01-01	1	500	2	Fechado
2	2016-05-09	2	300	2	Fechado
3	2015-06-11	2	1200	1	Aberto
4	2014-03-19	1	350	2	Fechado

11.2 – Operador CASE

- O código abaixo mostra uma forma mais simplificada de utilizar a estrutura do CASE para resolver o exemplo anterior:

```
SELECT id_pedido_venda, data, id_cliente, total,  
CASE id_status  
  WHEN 1 THEN 'Aberto'  
  WHEN 2 THEN 'Fechado'  
  ELSE 'Indefinido'  
END AS 'Status'  
FROM pedido_venda;
```

11.2 – Operador CASE

- O exemplo abaixo mostra que é possível colocar condições dentro da estrutura do CASE. Neste caso, se a data começar com 2016, resulta em 'Recente', se for entre 2015-12-01 e 2015-01-01, resultará em 'Ano Passado', senão resulta em 'Antigo':

```
SELECT id_pedido_venda, data, id_cliente, total, id_status,  
CASE  
  WHEN data LIKE '2016%' THEN 'Recente'  
  WHEN (data<='2015-12-01') AND (data>='2015-01-01') THEN 'Ano Passado'  
  ELSE 'Antigo'  
END AS 'Ano'  
FROM pedido_venda;
```

id_pedido_venda	data	id_cliente	total	id_status	Ano
1	2016-01-01	1	500	2	Recente
2	2016-05-09	2	300	2	Recente
3	2015-06-11	2	1200	1	Ano Passado
4	2014-03-19	1	350	2	Antigo

11.2 – Operador CASE

- O exemplo abaixo mostra uma consulta que retorna o id_pedido, data, nome do cliente, CPF/CNPJ e total. O CASE será usado para mostrar o CPF caso o cliente seja pessoa física, ou CNPJ no caso de pessoa jurídica:

```
SELECT id_pedido_venda, data, total, nome,  
CASE  
  WHEN cliente.tipo_cliente='F' THEN cpf  
  WHEN cliente.tipo_cliente='J' THEN cnpj  
  ELSE 'Indefinido'  
END AS 'Documento'  
FROM pedido_venda JOIN cliente USING(id_cliente);
```

id_pedido_venda	data	total	nome	Documento
1	2016-01-01	500	Paulo Soares	05699445977
2	2016-05-09	300	Fernando Prass	34554334567234
3	2015-06-11	1200	Fernando Prass	34554334567234
4	2014-03-19	350	Paulo Soares	05699445977

11.3 – Operador IFNULL

- Como mostrado no quadro abaixo, a função **IFNULL** possui dois argumentos, sendo que no caso do primeiro argumento ser NULL, será retornado o segundo argumento. No exemplo abaixo é exibido uma lista de clientes contendo o id_cliente, nome e documento (será mostrado o CPF ou o CNPJ caso o CPF seja null – nenhum cliente possui os dois campos preenchidos):

```
SELECT id_cliente, nome, tipo_cliente, IFNULL(cnpj, cpf) AS 'Documento' FROM cliente;
```

id_cliente	nome	tipo_cliente	Documento
1	Paulo Soares	F	05699445977
2	Fernando Prass	J	34554334567234

11.3 – Operador NULLIF

- A função **NULLIF** possui dois argumentos, sendo que no caso do primeiro argumento ser igual ao segundo, é retornado NULL. Mas caso o primeiro argumento seja diferente do segundo, é retornado o primeiro argumento:

//Como primeiro e segundo argumento são iguais retorna NULL

SELECT NULLIF(5, 5)

//Como primeiro e segundo argumento são diferentes retorna o primeiro argumento (5)

SELECT NULLIF(5, 6)