
# BANCOS DE DADOS RELACIONAIS

*Profº Me. Jones Artur Gonçalves*

# RELEMBRANDO CONCEITOS

| Cliente |
| --- |
| @Cod_cliente |
| Nome_cliente |
| Endereco |
| Cidade |
| Cep |
| UF |
| CPF |
| IE |
| |
| |
| |

| Pedido |
| --- |
| @Num_pedido |
| Prazo_entrega |
| Cod_cliente (E) |
| Cod_vendedor (E) |

| Item_pedido |
| --- |
| @ Num_pedido (E) |
| @ Cod_produto (E) |
| Quantidade |

| Vendedor |
| --- |
| @ Cod_vendedor |
| Nome_vendedor |
| Faixa_comissao |
| Salario_fixo |
| |

| Produto |
| --- |
| @ Cod_produto |
| Descricao |
| Unidade |
| Valor_unitario |
| |

Usaremos scripts para a criação destas tabelas

SQL

## ➢ Criação de Tabelas - Exemplo

Crie estas tabelas na nova base de dados  e inclua 5 registros em cada, para realização dos exemplos a seguir.

```
Create table Cidade
(Codcidade   int identity(1,1) not null primary key,
Nomecidade  varchar(40),
cdestado varchar(2));

Create table Empregado
(Nrmatricula int identity(1,1) primary key,
Nome        varchar(50),
Data_admissao datetime,
Salario     float)
```

# INSERT

➢ **Exemplo**

**INSERT INTO** cliente (cod_cliente, nome_cliente, endereco, cidade, cep, uf, cgc, ie) **VALUES** (0001,'Marcelo Cruz' ,'Rua Brasil, 35','Jundiaí',13124579,'SP', '7464563762','53425364');

OU

**INSERT INTO** cliente **VALUES** (0001,'Marcelo Cruz' ,'Rua Brasil, 35‘ ,'Jundiaí',13124579,'SP','7464563762', '53425364');

# CONSULTA A TABELAS

A estrutura básica de uma expressão SQL consiste de três cláusulas: **SELECT** , **FROM** e **WHERE** .

- **SELECT** - corresponde à operação projeção da álgebra relacional. É usada para listar os atributos desejados no resultado de uma consulta.

- **FROM** - corresponde à operação produto cartesiano da álgebra relacional. Ela lista as relações a ser examinadas na avaliação da expressão.

- **WHERE** - corresponde a predicado de seleção da álgebra relacional. Consiste em um predicado envolvendo atributos de relações que aparecem na cláusula **FROM** .

# SELECT

➢ **Sintaxe**

**SELECT** *nome_atributo1, nome_atributo2,...*
**FROM** *nome_tabela1, nome_tabela2...*
**WHERE** *condição*

A lista de atributos pode ser substituída por um asterisco (*) para selecionar todos os atributos de todas as relações presentes na cláusula FROM.

# SELECT

## ➢ Resumindo

O comando SELECT seleciona linhas e colunas de tabelas

- ❑ SELECT : especifica colunas
- ❑ FROM – especifica tabelas
- ❑ WHERE – especifica as linhas
- ❑ SELECT * recupera todas as colunas
- ❑ SELECT sem a cláusula WHERE recupera todas as linhas

# SELECT

## ➢ Usando Distinct

O comando DISTINCT suprime os itens duplicados de uma tabela

**Sintaxe** :

Select *|{[DISTINCT] coluna/expressão [apelido],...}
From tabela

# SELECT

## ➢ Usando Distinct

Exemplo:

Selecionando todas as colunas

Select * from cidade

| | codcidade | nomecidade | cdestado |
|---|---|---|---|
| 1 | 1 | Sorocaba | SP |
| 2 | 2 | São Roque | SP |
| 3 | 3 | Itú | SP |
| 4 | 5 | Guarapari | ES |

Selecionando apenas os estados das cidades cadastradas (sem repetição)

Select **distinct** cdestado from cidade

| | cdestado |
|---|---|
| 1 | ES |
| 2 | SP |

# SELECT

➢ **Exemplo**

Quais os tipos de unidades de produtos na tabela de produtos?

**SELECT** distinct unidade
**FROM** produto

# SELECT

## ➢ Selecionando apenas as colunas desejadas

Selecionando apenas as colunas desejadas

Select cdestado, nomecidade from cidade

| | cdestado | nomecidade |
|---|---|---|
| 1 | SP | Sorocaba |
| 2 | SP | São Roque |
| 3 | SP | Itú |
| 4 | ES | Guarapari |

# SELECT

- **Usando literal como atributo**

**Sintaxe** :

```
SELECT  'literal', nome_atributo1,....
        FROM nome_tabela
```

```
SELECT nome_atributo1  as  'literal',....
        FROM nome_tabela
```

# SELECT

➢ **Exemplo**

Listar o código e o nome de cada cliente

```
SELECT   'Código do Cliente', cod_cliente, 'Nome do Cliente',
nome_cliente
FROM cliente
==========================================

SELECT   cod_cliente as 'Código do Cliente', nome_cliente as 'Nome
do Cliente'
FROM cliente
```

# SELECT

## ➢ Selecionando colunas com expressão e apelido

Select nome, salario, salario+300 aumento_salario
from empregado

| | nome | salario | aumento_salario |
|---|---|---|---|
| 1 | Cris | 1000 | 1300 |
| 2 | Paula | 1100 | 1400 |
| 3 | Milton | 2000 | 2300 |
| 4 | Amanda | 5000 | 5300 |
| 5 | Fabio | NULL | NULL |

Obs.: Se qualquer valor da coluna em uma expressão aritmética for nulo, o resultado também será nulo.

# SELECT

## ➢ Concatenando valores (operador +)

```
Select 'Funcionario: '+nome+' Salário: '+CONVERT(VARCHAR, salario),
salario, nome
from empregado
```

| | (No column name) | salario | nome |
|---|---|---|---|
| 1 | Funcionario: Cris Salário: 1000 | 1000 | Cris |
| 2 | Funcionario: Paula Salário: 1100 | 1100 | Paula |
| 3 | Funcionario: Milton Salário: 2000 | 2000 | Milton |
| 4 | Funcionario: Amanda Salário: 5000 | 5000 | Amanda |
| 5 | NULL | NULL | Fabio |

# SELECT

## ➢ Tipos de Condições Permitidas

| Operações | Operadores |
|---|---|
| Relacionais | = > < >= <= <> != !< !> |
| Faixa de valores | BETWEEN e NOT BETWEEN |
| Comparação com cadeia de caracteres | LIKE e NOT LIKE |
| Pertinência a conjuntos | IN e NOT IN |
| Valores desconhecidos | IS NULL e IS NOT NULL |
| Combinação de operações | AND, OR |
| Negações | NOT |

# SELECT

## ➢ Operações Relacionais

**Sintaxe** :

```
SELECT nome_atributo1, nome_atributo2....
        FROM tabela
        WHERE  expressão operador_de_comparação expressão
```

# SELECT

➢ **Exemplo**

Listar o número do pedido , o código do produto e a quantidade dos itens do pedido com quantidade igual a 45.

```
SELECT num_pedido, cod_produto, quantidade
FROM item_pedido
WHERE quantidade = 45
```

# SELECT

➢ **Exemplo**

Listar todos os funcionários que trabalham no setor de código 2.

```
Select nrmatricula, primeiro_nome, ultimo_nome, cod_setor
from funcionario
where cod_setor = 2
```

# SELECT

➢ **Exemplo**

Listar todos os funcionários que recebem salário menor que 1500

```
Select primeiro_nome, salario
from funcionario
where salario < 1500
```

# SELECT

- **Operações baseada em faixas de valores**

**Sintaxe** :

```
SELECT nome_atributo1, nome_atributo2....
        FROM tabela
        WHERE  expressão [NOT] BETWEEN expressão AND expressão
```

# SELECT

➢ **Exemplo**

Listar o nome e salário dos funcionários que recebem entre 1500 e 2000

```
Select primeiro_nome, salario
from funcionario
where salario between 1500 and 2000
```

limite inferior  limite superior

# SELECT

➢ **Exemplo**

Listar o código e a descrição dos produtos que tenham o valor unitário na faixa R$0,32 até R$2,00

```
SELECT cod_produto, descricao
FROM produto
WHERE valor_unitario between 0.32 and 2.00
```

# SELECT

## ➢ Comparações entre Cadeias de Caracteres

**Sintaxe** :

```
SELECT nome_atributo1,  Nome_atributo2....
        FROM tabela
        WHERE  expressão [NOT] LIKE  'cadeia de caracteres'
```

# SELECT

## ➢ Comparações entre Cadeias de Caracteres

O operador LIKE é utilizado para comparações em cadeias de caracteres. Os *padrões* são descritos usando os seguintes caracteres especiais:

- ❖ por cento (**%**) - O caractere % substitui qualquer subcadeia.
- ❖ sublinhado ( _ ) - O caractere _ substitui qualquer caractere.
- ❖ Caracteres entre colchetes (**[]**) - Qualquer caracter dentro da faixa ou conjunto especificado. (SQL Server)
- ❖ **[^] -** Qualquer caracter que não esteja dentro da faixa ou conjunto especificado. (SQL Server)

# SELECT

## ➢ Exemplos de utilização de LIKE

| **Expressão** | **Resultado** |
|---|---|
| LIKE ‘BR%’ | Nomes que comecem com **BR** |
| LIKE ‘%een’ | Nomes que terminem com **een** |
| LIKE ‘%en%’ | Nomes que tenham a seqüência **en** em qualquer posição |
| LIKE ‘_en’ | Nomes de três letras terminando por **en** |
| LIKE ‘[CK]%’ | Nomes que comecem com **C** ou **K** |
| LIKE ‘[S-V]ing’ | Nomes de quatro letras que terminem **ing** e comecem com uma letra de **S** a **V** |
| LIKE ‘M[^c]%’ | Nomes que comecem com **M** e não tenham a letra **c** como segundo caracter |

# SELECT

➢ **Exemplo**

Listar todos os funcionários que tenham a letra M em qualquer parte do nome

**Select** primeiro_nome, salario, cod_setor
**from** funcionario
**where** primeiro_nome **like** '%M%'

| | primeiro_nome | salario |
|---|---|---|
| 1 | Yasmin | 5000 |
| 2 | Maria Aparecida | 2500 |
| 3 | Marcia | 3000 |
| 4 | Milton | 2000 |

# SELECT

➢ **Exemplo**

Listar todos os funcionários que tenham a letra M em qualquer parte do nome

```
Select primeiro_nome, salario, cod_setor
from funcionario
where primeiro_nome like 'M%'
```

| | primeiro_nome | salario |
|---|---|---|
| 1 | Maria Aparecida | 2500 |
| 2 | Marcia | 3000 |
| 3 | Milton | 2000 |

# SELECT

➢ **Exemplo**

Listar todos os produtos que tenham a sua unidade começando por K

```
SELECT cod_produto, descricao, unidade
FROM produto
WHERE unidade like 'K%'
```

# SELECT

➢ **Complete a tabela com o resultado da expressão.**

| **Expressão** | **Resultado** |
|---|---|
| LIKE 'Juca%' | |
| LIKE '%Silva' | |
| LIKE '%Santos%' | |
| LIKE 'A_' | |
| LIKE '_A' | |
| LIKE '_A_' | |
| LIKE '%A_' | |
| LIKE '_A%' | |
| LIKE '___' | |
| LIKE '___%' | |
| LIKE '%‘'%' | |

# SELECT

## ➢ Complete a tabela com o resultado da expressão.

| Expressão | Resultado |
|---|---|
| LIKE 'Juca%' | Qualquer string que iniciem com Juca. |
| LIKE '%Silva' | Qualquer string que terminem com Silva. |
| LIKE '%Santos%' | Qualquer string que tenha Santos em qualquer posição. |
| LIKE 'A_' | String de dois caracteres que tenham a primeira letra A e o segundo caractere seja qualquer outro. |
| LIKE '_A' | String de dois caracteres cujo primeiro caractere seja qualquer um e a última letra seja a letra A. |
| LIKE '_A_' | String de três caracteres cuja segunda letra seja A, independentemente do primeiro ou do último caractere. |
| LIKE '%A_' | Qualquer string que tenha a letra A na penúltima posição e a última seja qualquer outro caractere. |
| LIKE '_A%' | Qualquer string que tenha a letra A na segunda posição e o primeiro caractere seja qualquer outro caractere. |
| LIKE '___' | Qualquer string com exatamente três caracteres. |
| LIKE '___%' | Qualquer string com pelo menos três caracteres. |
| LIKE '%''%' | Qualquer string que tenha o caractere ' em qualquer posição. |

# SELECT

- **Operações de pertinência a Conjuntos ( Listas)**

**Sintaxe** :

```
SELECT nome_atributo1, nome_atributo2....
        FROM tabela
        WHERE [NOT] expressão [NOT] IN   (lista de valores)
```

# SELECT

➢ **Exemplo**

Listar os vendedores que têm a faixa de comissão A ou B

```
SELECT nome_vendedor
FROM vendedor
WHERE faixa_comissao in ('A', 'B')
```

# SELECT

➢ **Exemplo**

Listar o nome, salário e setor de todos os funcionários que trabalham nos departamentos de códigos 1 e 3

```
Select primeiro_nome, salario, cod_setor
from funcionario
where cod_setor in (1,3)
```

# SELECT

➢ **Exemplo**

Listar o nome, salário e setor de todos os funcionários que trabalham nos departamentos de códigos 1 e 3

```
Select *
from funcionario f
where cod_setor NOT IN(1,3)
```

**Usando o operador NOT**

# SELECT

- **Operações com valores desconhecidos**

**Sintaxe** :

```
SELECT nome_atributo1, nome_atributo2....
        FROM tabela
        WHERE nome_atributo IS [NOT] NULL
```

# SELECT

➢ **Exemplo**

Mostrar os clientes que **não** tenham inscrição estadual

```
SELECT *
FROM cliente
WHERE ie is null
```

# SELECT

➢ **Exemplo**

Mostrar os clientes que tenham inscrição estadual

```
SELECT *
FROM cliente
WHERE ie is not null
```

# SELECT

➢ **Exemplo**

Mostrar os clientes que **não** tenham e-mail

**Select** primeiro_nome, salario, cod_setor
**from** funcionario
**where** email **is null**


➢ **Exemplo**

Mostrar os clientes que tenham e-mail

```
Select primeiro_nome, salario, cod_setor
from funcionario
where email is not null
```

# SELECT

- **Combinação de operações**

**Sintaxe** :

```
SELECT nome_atributo1, nome_atributo2....
        FROM tabela
        WHERE [NOT] EXPRESSÃO {AND/ OR} [NOT] expressão
```

# SELECT

➢ **Exemplo**

Mostrar todos os funcionários que tem um salário maior que 1000 e matrícula maior ou igual a 3

```
Select nrmatricula, ultimo_nome, salario
from funcionario
where salario > 1000 and nrmatricula >= 3
```

**AND exige que ambas as condições sejam verdadeiras**

# SELECT

➢ **Exemplo**

Mostrar todos os funcionários que tem um salário maior que 2500 ou código de setor igual a TI

```
Select *
from funcionario
where salario > 2500 or cod_setor = 4
```

**OR exige que uma das condições sejam verdadeiras**

# SELECT

- **Exemplo**

Listar os produtos que tenham unidade igual a ‘M’ e valor unitário igual a R$ 1,05

```
SELECT descricao
FROM produto
WHERE unidade = 'M' and valor_unitario  = 1.05
```

Listar os produtos que tenham unidade igual a ‘UN’ e valor unitário igual a R$ 4,00

```
SELECT descricao
FROM produto
WHERE unidade = 'UN' and valor_unitario  = 4
```

# SELECT

- **Valores duplicados**

**Sintaxe** :

```
SELECT [ALL/DISTINCT] nome_atributo1, nome_atributo2....
        FROM tabela
        WHERE  condições
```

# SELECT

- **Classificação dos dados**

**Sintaxe** :

```
SELECT nome_atributo1, nome_atributo2....
        FROM tabela
        WHERE  condições
        [ORDER BY  nome_atributon / número_da_coluna
        [ASC / DESC]]....
```

# SELECT

➢ **Exemplo**

Listar em ordem alfabética a lista de vendedores e seus respectivos salários fixos.

```
SELECT nome_vendedor, salario_fixo
FROM vendedor
ORDER by nome_vendedor
```

**ASC: ordem crescente, default**

# SELECT

➢ **Exemplo**

Listar em ordem alfabética a lista de funcionários por ordem de setor

```
Select *
from funcionario f
where cod_setor NOT IN (2,4)
Order by cod_setor
```

**ASC: ordem crescente, default**

# SELECT

➢ **Exemplo**

Listar em ordem alfabética a lista de funcionários por ordem decrescente de setor

```
Select *
from funcionario f
where cod_setor NOT IN (2,4)
Order by cod_setor DESC
```

**DESC: ordem decrescente**

# SELECT

➢ **Exemplo**

Listar em ordem alfabética a lista de funcionários por ordem de setor e primeiro nome

```
Select *
from funcionario f
where cod_setor NOT IN (2,4)
Order by cod_setor, primeiro_nome
```

**Classificando por várias colunas**

# INSERT com SELECT

Inserir valores em uma tabela provenientes do select de outra já existente:

Exemplo:

```
INSERT INTO EMPREGADOS(CODIGO,NOME, SALARIO, SECAO)
    SELECT CODIGO,NOME,SALARIO, SECAO
FROM EMPREGADOS_FILIAL
WHERE DEPARTAMENTO = 2
```

# INSERT com SELECT

Exemplo:

```
CREATE TABLE PESSOA
(
 id_pessoa integer primary key,
 nome varchar(20),
 cpf varchar(14)
);
```

```
CREATE TABLE PESSOA_FISICA
(
 id_pessoa integer primary key,
 nome varchar(20),
 cpf varchar(14)
);
```

```
INSERT INTO PESSOA VALUES (1, ‘PEDRO CABRAL', '12345678991');
```

```
INSERT INTO PESSOA_FISICA SELECT ID_PESSOA, NOME, CPF FROM PESSOA;
```

# Atividade de Fixação

1. Resolver Lista de exercícios

# BIBLIOGRAFIA

## BÁSICA:


DATE, C. J. PROJETO DE BANCO DE DADOS E TEORIA RELACIONAL: FORMAS NORMAIS E TUDO O MAIS. SÃO PAULO: NOVATEC, 2015.

ELMASRI, R.; NAVATHE, S. B. SISTEMAS DE BANCO DE DADOS: FUNDAMENTOS E APLICAÇÕES. 7 ED. SÃO PAULO: PEARSON, 2019.

HEUSER, C. A. PROJETO DE BANCO DE DADOS. 6 ED. PORTO ALEGRE: BOOKMAN, 2010.


## COMPLEMENTAR:


HARRINGTON, J. L. Projeto de Bancos de Dados Relacionais: Teoria e Prática. São Paulo: Campus, 2002.

MACHADO, F. N. R., Banco de dados: projeto e implementação. 2 ed. São Paulo: Érica, 2008.

NADEAU, Tom et al. Projeto e Modelagem de Banco de Dados. 5 ed. Rio de Janeiro: Elsevier Brasil, 2013.

SILBERSCHATZ, Abraham; SUNDARSHAN, S.; KORTH, Henry F. Sistema de banco de dados. Rio de Janeiro: Elsevier Brasil, 2016.

# Referências


ALVES, W. P. FUNDAMENTOS DE BANCOS DE DADOS. ÉRICA, 2004

HEUSER, CARLOS ALBERTO. PROJETO DE BANCO DE DADOS. SAGRA LUZZATTO, 2004.

TEOREY, TOBY J. PROJETO E MODELAGEM DE BANCO DE DADOS. ELSEVIER, 2007.

O.K. TAKAI; I.C.ITALIANO; J.E. FERREIRA, INTRODUÇÃO A BANCO DE DADOS

OSVALDO KOTARO, APOSTILA, DCC-IME-USP – FEVEREIRO - 2005

MATTOSO, MARTA, INTRODUÇÃO À BANCO DE DADOS – AULA

GILLENSON, MARK L. FUNDAMENTOS DE SISTEMAS DE GERÊNCIA DE BANCO DE DADOS. LTC, 2006.

BANCO DE DADOS BÁSICO, UNICAMP, CENTRO DE COMPUTAÇÃO, SLIDES.

BOGORNY VANIA, MODELO ENTIDADE-RELACIONAMENTO, SLIDES.

WWW.JOINVILLE.UDESC.BR/PORTAL/PROFESSORES/MAIA/.../6___MODELO_ER.PPT DATA DE ACESSO: 01/07/2015

ABREU, FELIPE MACHADO; ABREU, MAURÍCIO – PROJETO DE BANCO DE DADOS – UMA VISÃO PRÁTICA - ED. ÉRICA – SÃO PAULO

HEUSER, CARLOS ALBERTO. PROJETO DE BANCO DE DADOS – UMA VISÃO PRÁTICA. PORTO ALEGRE: SAGRA LUZATTO, 2004.

KORTH, H. F.; SUDARSHAN, S; SILBERSCHATZ, A. SISTEMA DE BANCO DE DADOS. 5A ED. EDITORA CAMPUS, 2006. - CAPÍTULO 6

HTTP://WWW.PROFTONINHO.COM/DOCS/MODELAGEM_AULA_6_ENTID_ASSOC.PDF DATA DE ACESSO: 01/07/2015

HTTPS://MATERIALPUBLIC.IMD.UFRN.BR/CURSO/DISCIPLINA/4/56/1/6 DATA DE ACESSO: 01/02/2023

ELMASRI, R.; NAVATHE S. B. SISTEMAS DE BANCO DE DADOS. 4 ED. EDITORA ADDISON-WESLEY. 2005. - CAPÍTULO 3

DAVENPORT, THOMAS H.; PRUSAK, LAURENCE. CONHECIMENTO EMPRESARIAL: COMO AS ORGANIZAÇÕES GERENCIAM O SEU CAPITAL INTELECTUAL. RIO DE JANEIRO: CAMPUS, 1998.

HTTP://WWW.IME.UNICAMP.BR/~HILDETE/DADOS.PDF ACESSO EM: 12 MAIO 2016.

# OBRIGADO