



_Apresentações

LET'S CODE

| _Michael Tadeu



Formado em Sistemas de Informação pela Ufla, Mestre em Engenharia de Software e Banco de dados pela Ufla e pós-graduação em Arquitetura de Software Distribuído pela PucMinas. Desenvolvedor Sênior Full Stack na Axxiom Tecnologia e Inovação e Professor no Degree Web Full.



_Banco de Dados

LET'S CODE

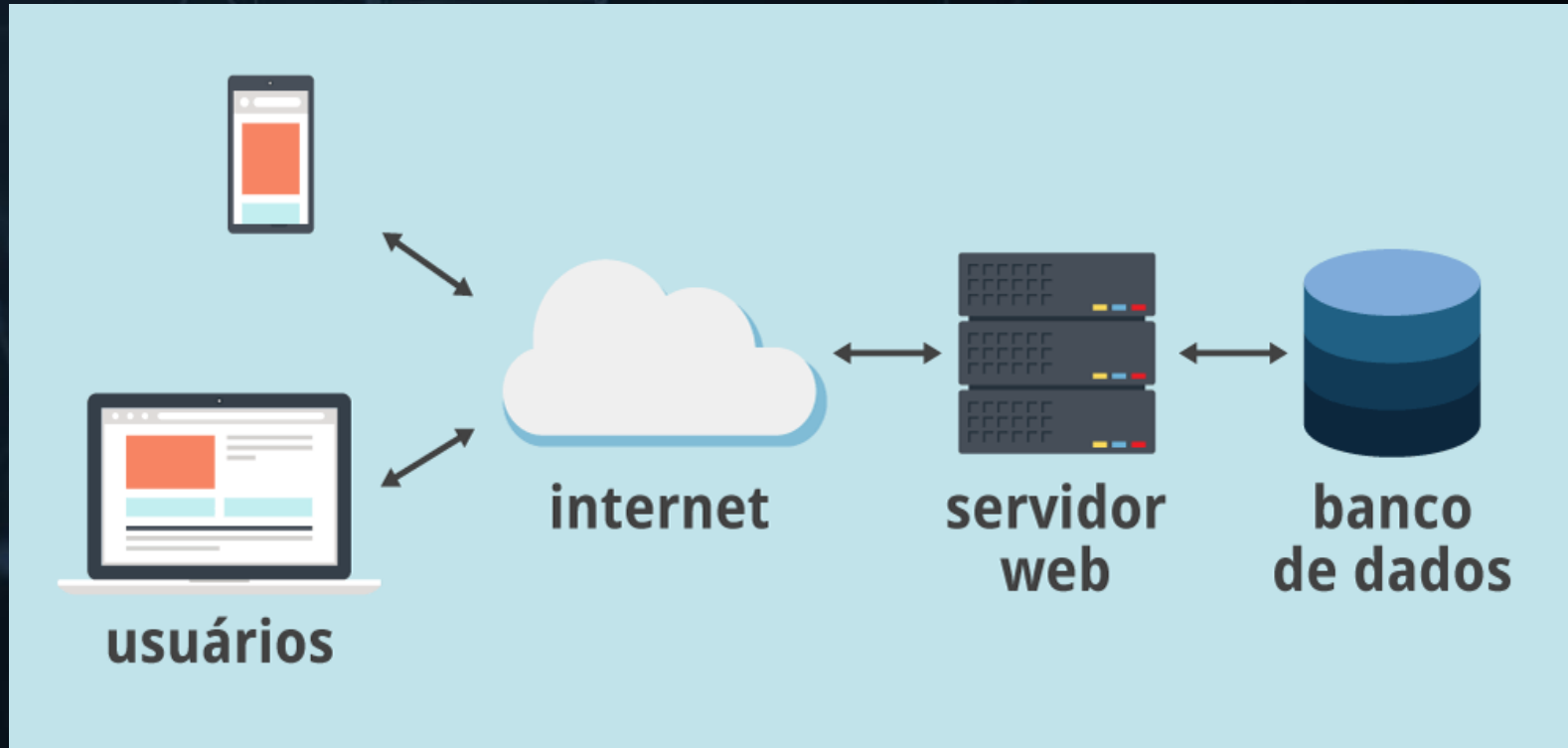
| _O que Vamos Aprender?

- Introdução;
- Modelagem Entidade-Relacionamento (MER);
- Modelo Físico e Normalização;
- Queries Simples;
- Queries Complexas;
- Otimização.

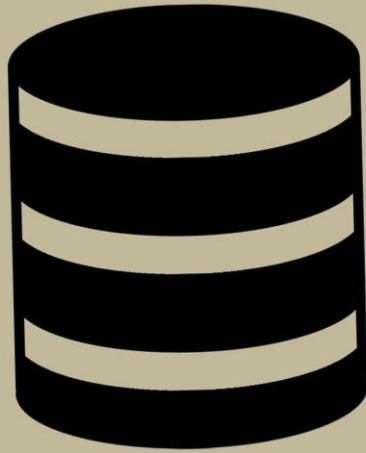
| _Introdução a Banco de Dados

"O principio básico de um Banco de Dados é armazenar informações de um sistema."

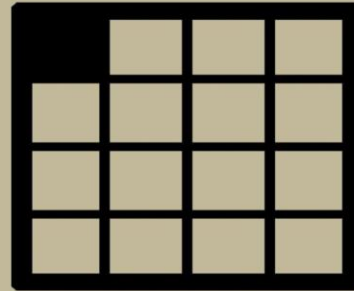
| _Introdução a Banco de Dados



| _Diferença entre um banco de dados e uma planilha?



Vs.



| _O que é Structured Query Language (Linguagem de consulta estruturada)?



| _Tipos de Bancos de Dados - Relacional

ORACLE®



Microsoft®
SQL Server®



DB2®



PostgreSQL



SQLite



MySQL®

| _SQL – Banco de dados Relacional

```
SELECT * FROM usuario WHERE  
estado = "São Paulo"
```

| _Tipos de Bancos de Dados - NoSQL



Cassandra

mongoDB



membase



| _Chave e Valor ou Orientado a Documento – Banco de dados NoSQL

```
db.usuarios.find(  
  { estado: { $eq: "São Paulo" } }  
)
```

|_SGBDs



|_Qual utilizaremos?



| _Vamos instalar





_Banco de Dados

LET'S CODE

| _Resolvendo Problemas



| _O que Vamos Aprender?

- Introdução;
- Modelagem Entidade-Relacionamento (MER);
- Modelo Físico e Normalização;
- Queries Simples;
- Queries Complexas;
- Otimização.

| _Tipos de Dados - Numéricos

- inteiros;
- smallint;
- bigint.
- decimais;
- real;
- double precision.

| _Tipos de Dados - Textos

- character ou char;
- character varying ou varchar;
- text.

| _Tipos de Dados - Data

- date;
- time;
- timestamp.

| _Tipos de Dados – Tipos Lógicos

- boolean;
- bit.

| _Tipos de Dados – Enumerados

- enum.

| _Tipos de Dados – Outros

- geográfico;
- monetário;
- endereço de rede;
- bit string;
- text search;
- xml;
- json;
- arrays;
- composite range.

| _Criar Tabelas e seus Tipos de Dados

CLIENTE (cod_cli, nome_cli, endereco, cidade, cep, uf)

VENDEDOR (cod_vend, nome_vend, sal_fixo, faixa_comiss)

PEDIDO (num_ped, prazo_entr, cd_cli, cd_vend)

ITEM_PEDIDO (no_ped, cd_prod, qtd_ped)

PRODUTO (cod_prod, unid_prod, desc_prod, val_unit)



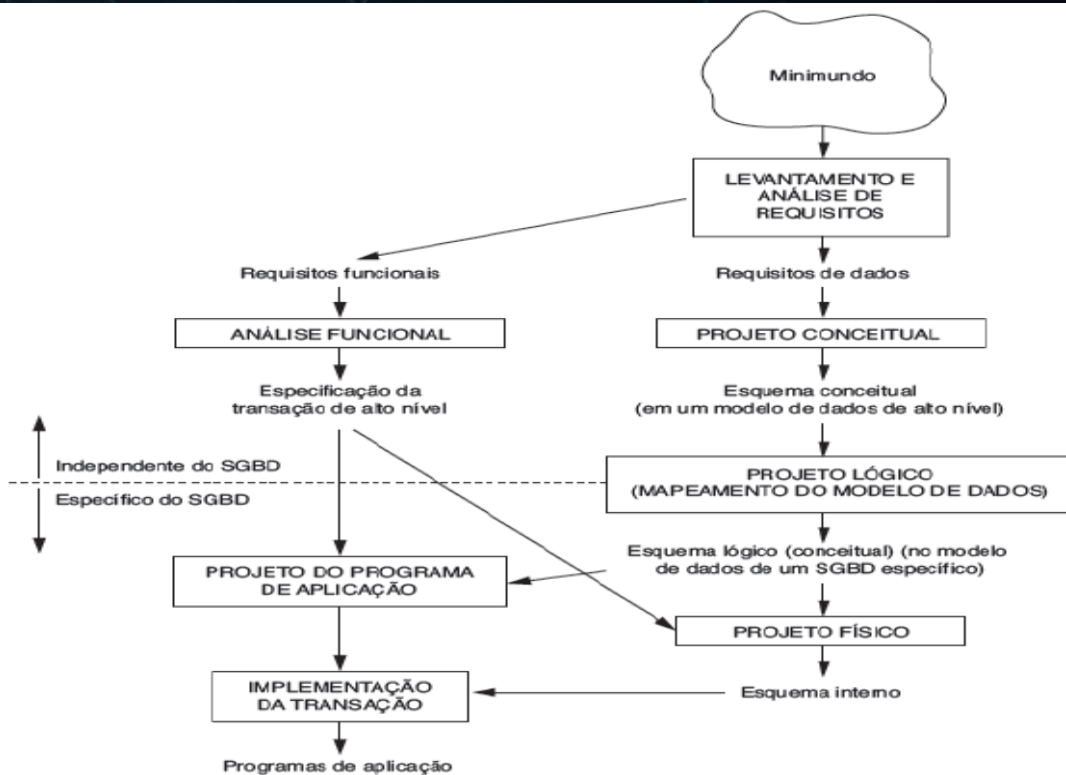
_Banco de Dados

LET'S CODE

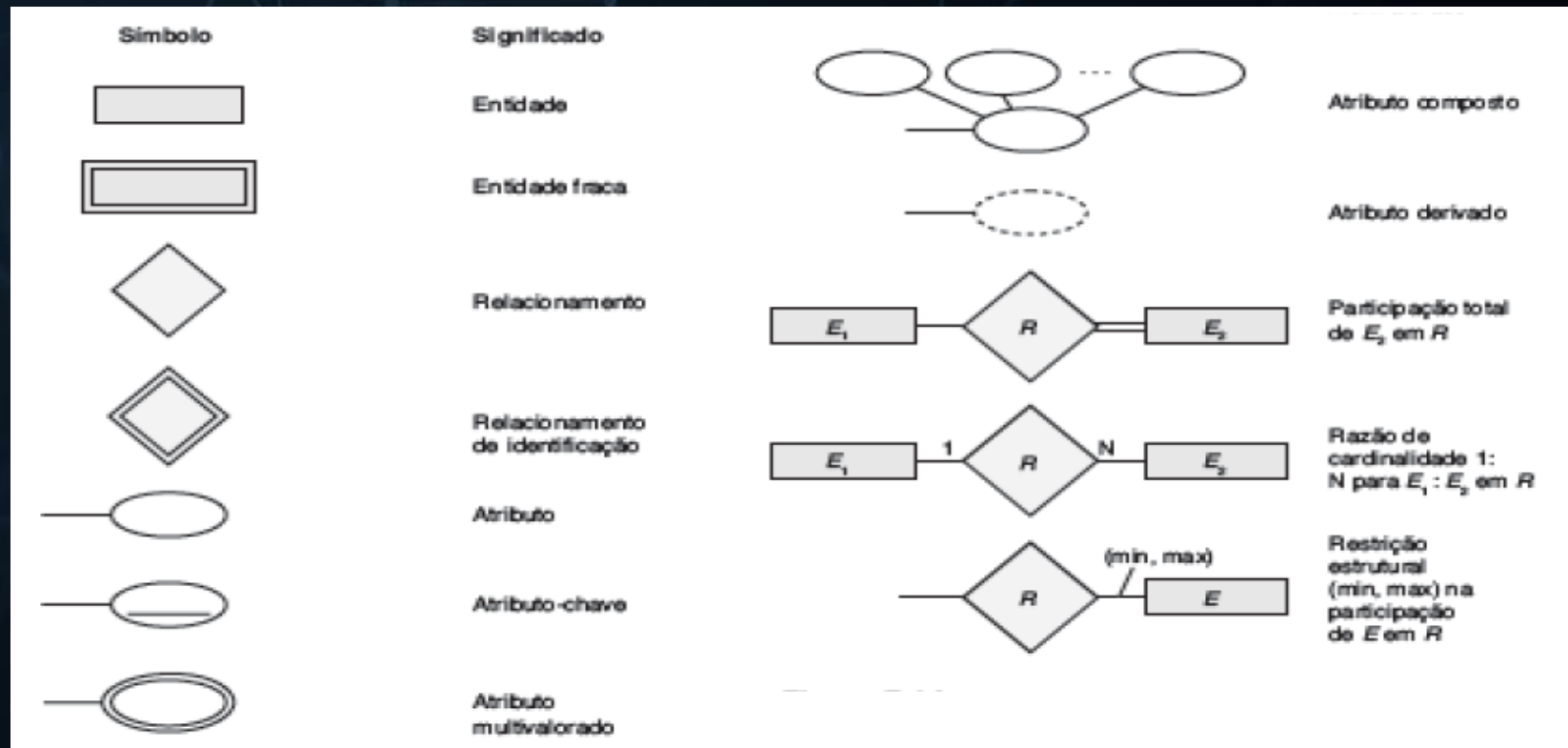
| _Dúvidas



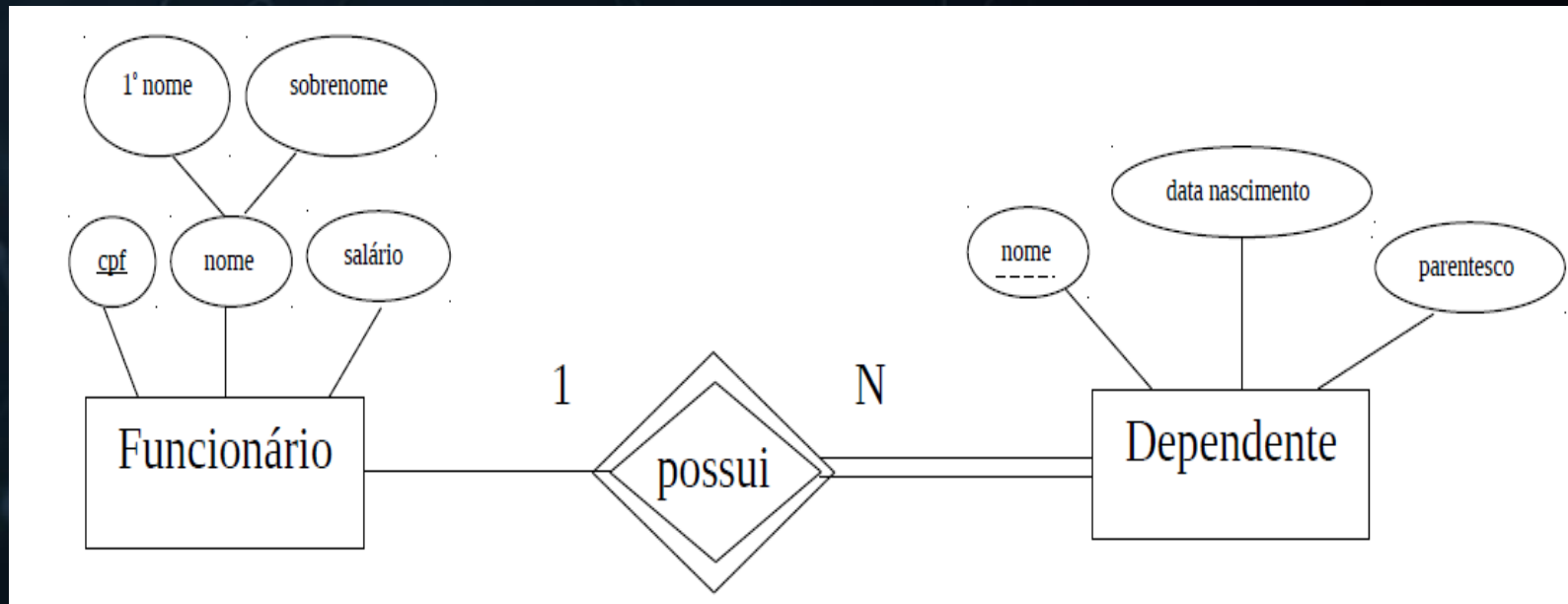
_Modelagem de Dados



|_Modelagem de Dados - MER



| _Modelagem de Dados - MER



| _Modelagem de Dados - Cardinalidade

1:1 - Um para Um.



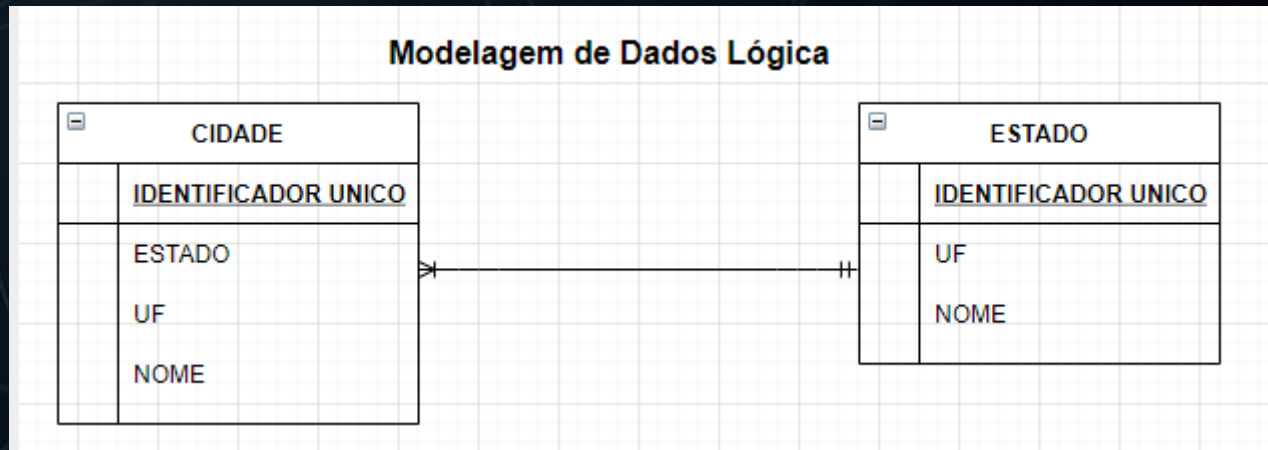
1:N - Um para Muitos.



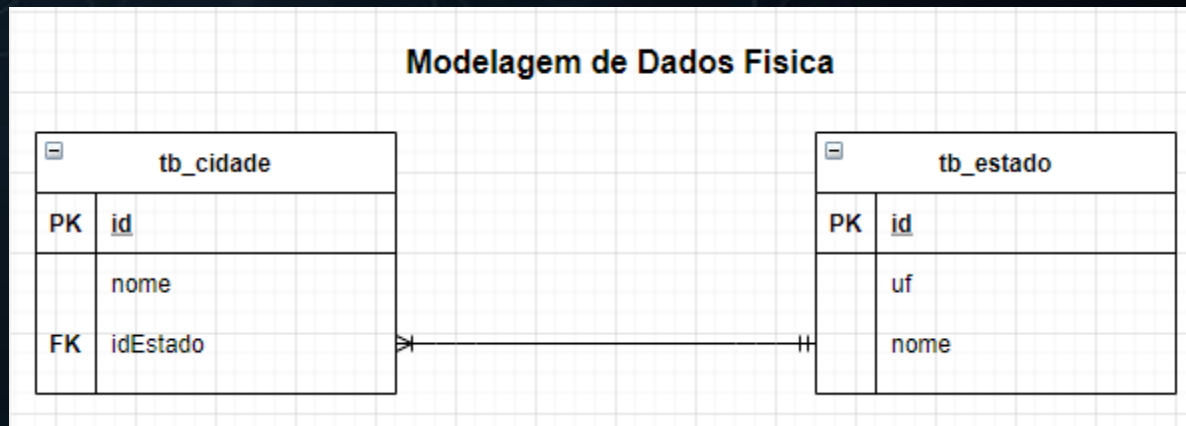
M:N - Muitos para Muitos.



| _Modelagem de Dados Lógica



| _Modelagem de Dados Física



| _Modelagem de Dados

| Característica | Conceitual | Logica | Física |
|------------------------------|------------|--------|--------|
| Nome de Entidades | X | X | |
| Relacionamentos de Entidades | X | X | |
| Atributos | X | X | |
| Chave Primária (PK) | | X | X |
| Chave Estrangeira (FK) | | X | X |
| Nome das Tabelas | | | X |
| Nome das Colunas | | | X |
| Tipo das Colunas | | | |

| _Criar Tabelas e seus Tipos de Dados

Você deverá criar a modelagem para o catálogo de filmes, observando a seguinte situação:

- um filme possui apenas um único gênero;
- um gênero pode definir mais de um filme.



_Banco de Dados

LET'S CODE

| _Dúvidas



|_Constraints – NOT NULL

```
CREATE TABLE cliente (  
    cod_cli INTEGER NOT NULL,  
    cpf CHAR(11) UNIQUE,  
    nome_cli VARCHAR(40) NOT NULL,  
    idade INTEGER CHECK (idade = 18)  
    endereco VARCHAR(40) null,  
    cidade VARCHAR(20) null,  
    cep CHAR(8) null,  
    uf CHAR(2) null,  
    ativo BIT null  
);
```

|_Constraints – UNIQUE

```
CREATE TABLE cliente (  
    cod_cli INTEGER NOT NULL,  
    cpf CHAR(11) UNIQUE,  
    nome_cli VARCHAR(40) NOT NULL,  
    idade INTEGER CHECK (idade = 18)  
    endereco VARCHAR(40) null,  
    cidade VARCHAR(20) null,  
    cep CHAR(8) null,  
    uf CHAR(2) null,  
    ativo BIT null  
);
```

|_Constraints – DEFAULT

```
CREATE TABLE conta (  
    cod_conta BIGINT NOT NULL,  
    saldo DOUBLE DEFAULT 0.0  
);
```


|_Constraints – CHECK

```
CREATE TABLE cliente (  
    cod_cli INTEGER NOT NULL,  
    cpf CHAR(11) UNIQUE,  
    nome_cli VARCHAR(40) NOT NULL,  
    idade INTEGER CHECK (idade = 18)  
    endereco VARCHAR(40) null,  
    cidade VARCHAR(20) null,  
    cep CHAR(8) null,  
    uf CHAR(2) null,  
    ativo BIT null  
);
```

| _Constraints – CHAVES PRIMARIAS

```
CREATE TABLE letscode_genero (  
    id_genero INTEGER NOT NULL,  
    nome VARCHAR(40) NOT NULL,  
    PRIMARY KEY (id_genero)  
);
```

| _Constraints – CHAVES ESTRANGEIRAS

```
CREATE TABLE letscode_participacao (  
    id_participacao INTEGER NOT NULL,  
    id_filme INTEGER NOT NULL,  
    id_ator INTEGER NOT NULL,  
    FOREIGN KEY (id_filme)  
        REFERENCES letscode_filme (id_filme),  
    FOREIGN KEY (id_ator)  
        REFERENCES letscode_ator (id_ator),  
    PRIMARY KEY (id_participacao)  
);
```

| _Funções Agregadas - MIN() e MAX()

```
SELECT MIN(valor_unitario) FROM tb_entrada_produto;
```

```
SELECT MAX(valor_unitario) FROM tb_entrada_produto;
```

| _Funções Agregadas - COUNT()

```
SELECT COUNT(*) FROM tb_saida_produto;
```

| _Funções Agregadas - SUM()

```
SELECT qtde, SUM(valor_unitario) FROM tb_saida_produto  
GROUP BY 1;
```

| _Funções Agregadas - AVG()

```
SELECT valor_unitario, AVG(qtde) FROM tb_entrada_produto  
GROUP BY 1;
```

| _Funções Agregadas - HAVING

```
SELECT valor_unitario, COUNT(*) FROM tb_estoque  
GROUP BY 1  
HAVING COUNT(*) > 2;
```


|_Consulta

Consultas

Na tabela Customers:

1) Gere uma relação com os nomes dos clientes, suas cidades e países, em ordem alfabética de nome.



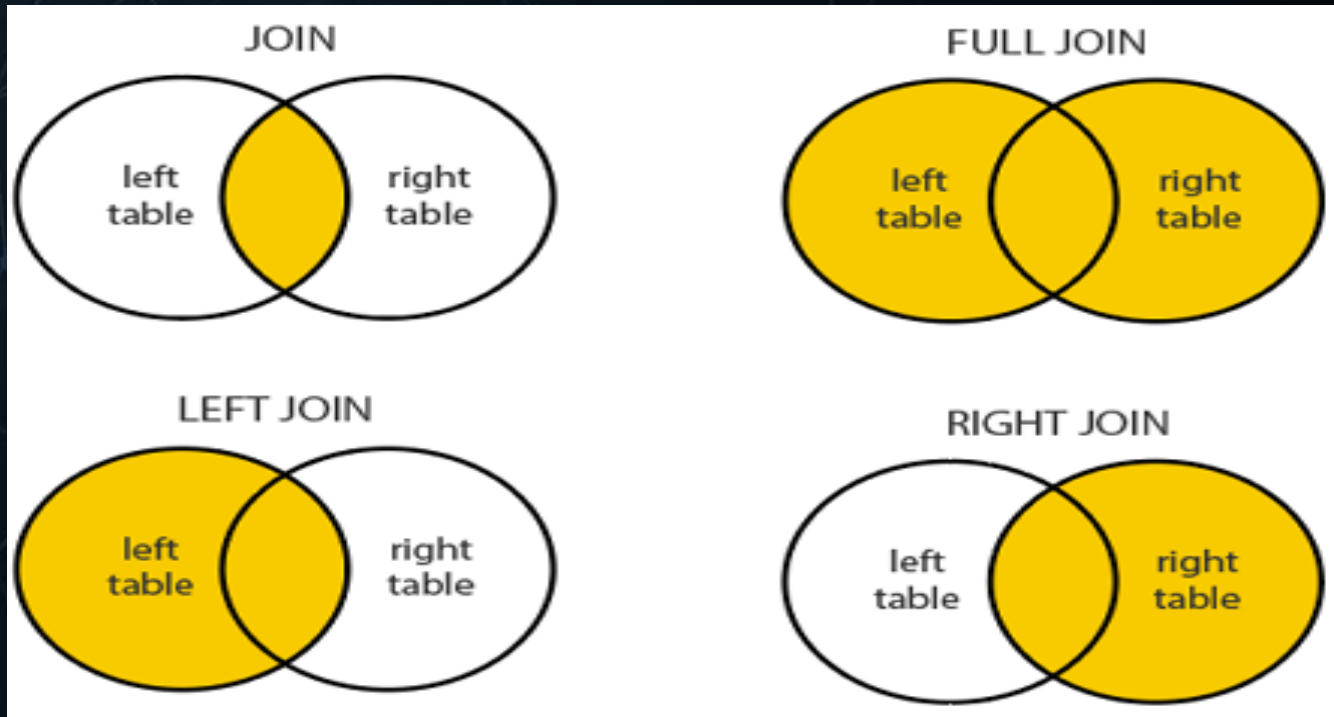
_Banco de Dados

LET'S CODE

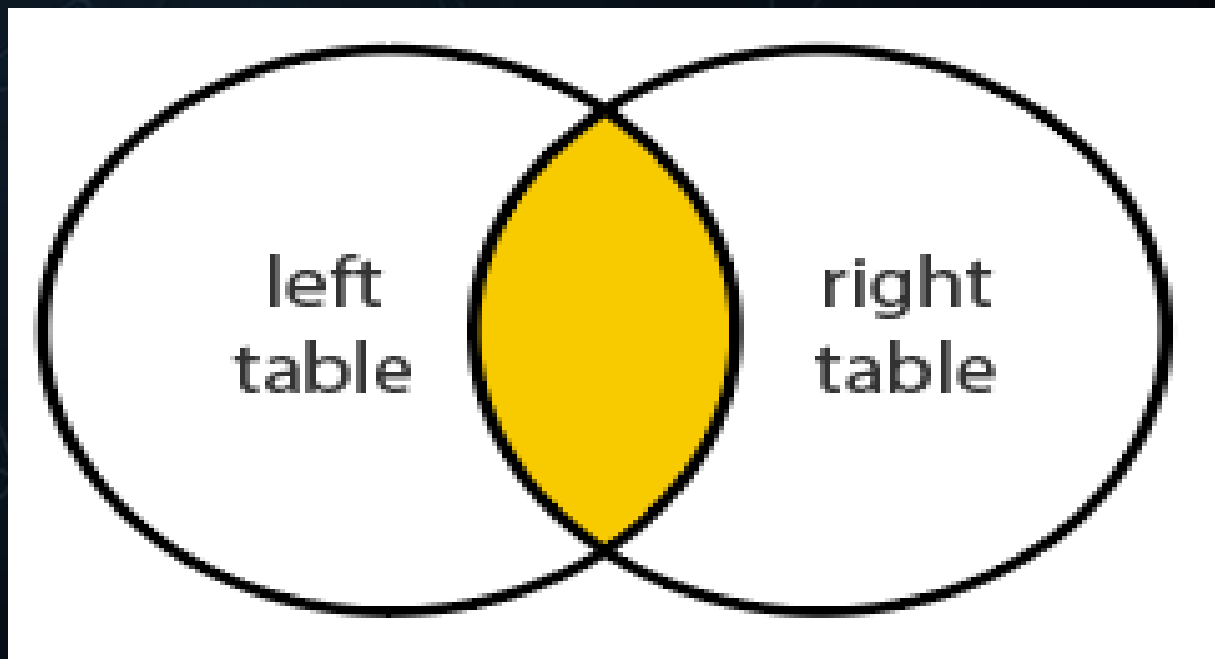
| _Dúvidas



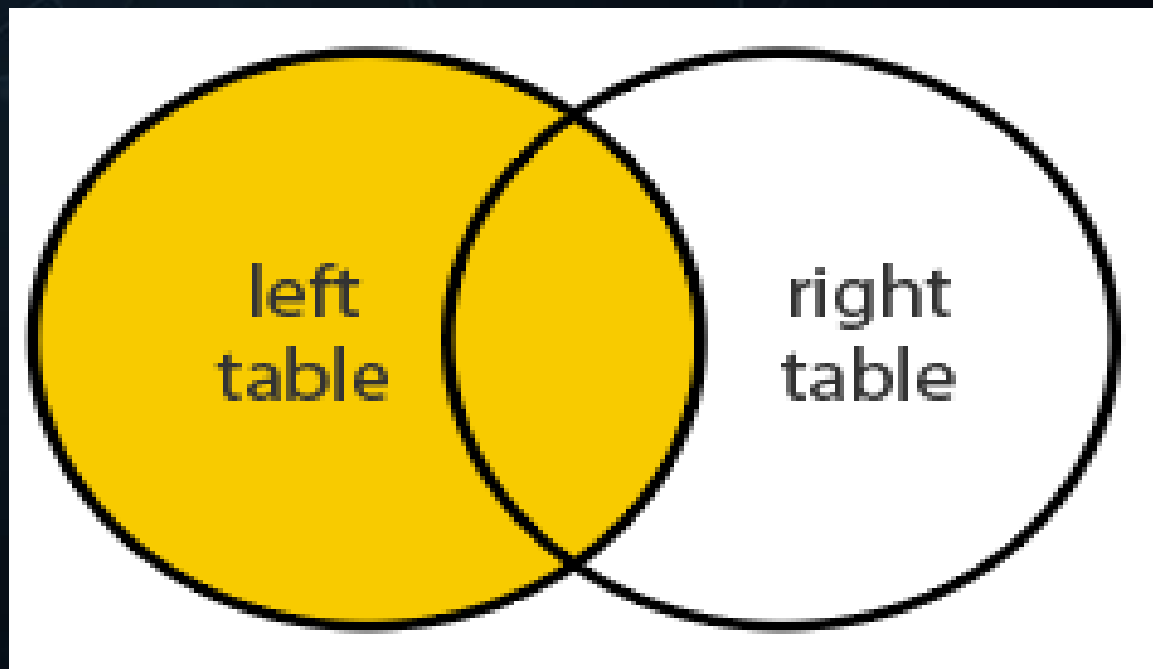
| _Introdução a 'Otimização de Consultas no Banco de Dados'



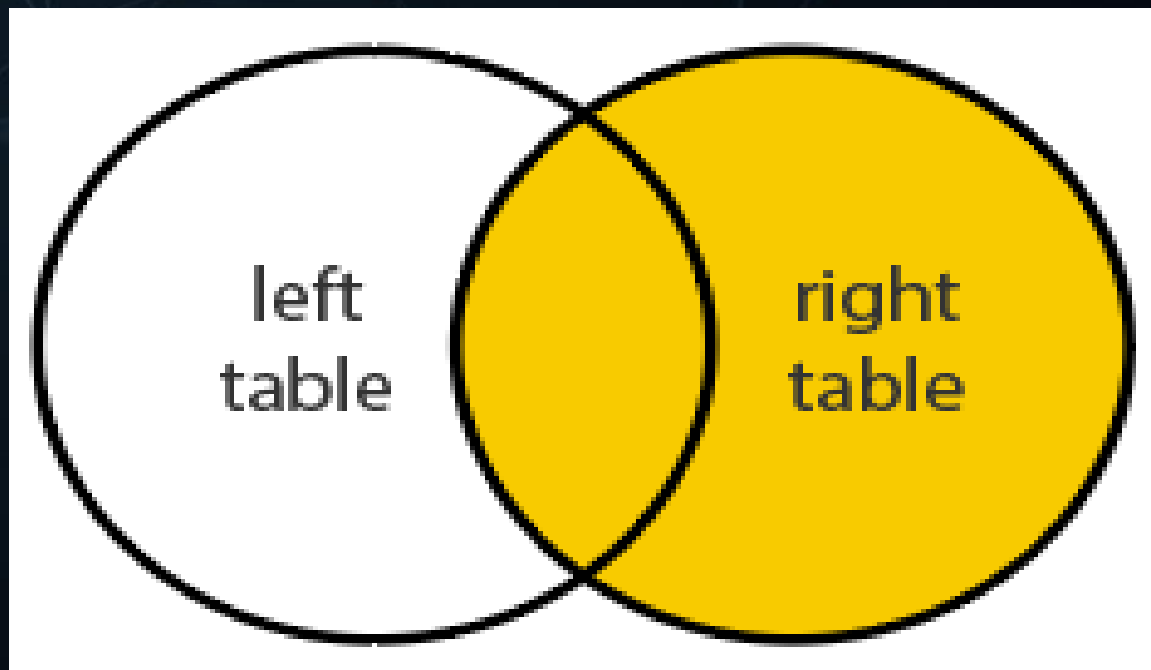
| INNER JOIN



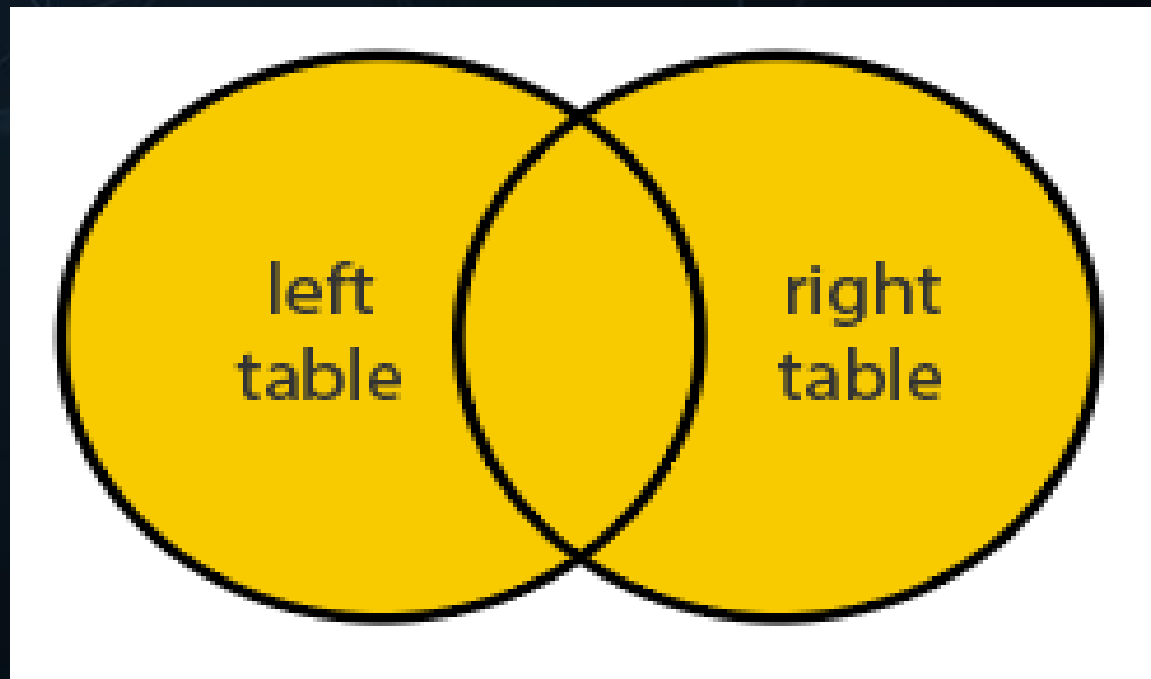
| _LEFT JOIN



|_RIGHT JOIN



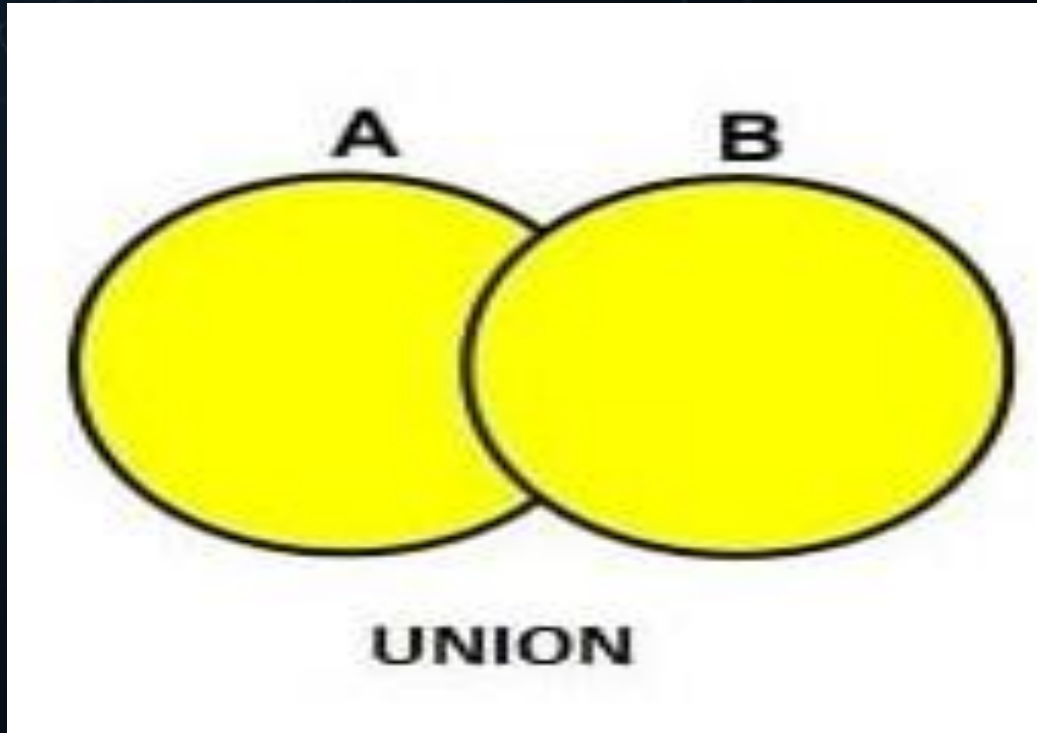
|_FULL JOIN



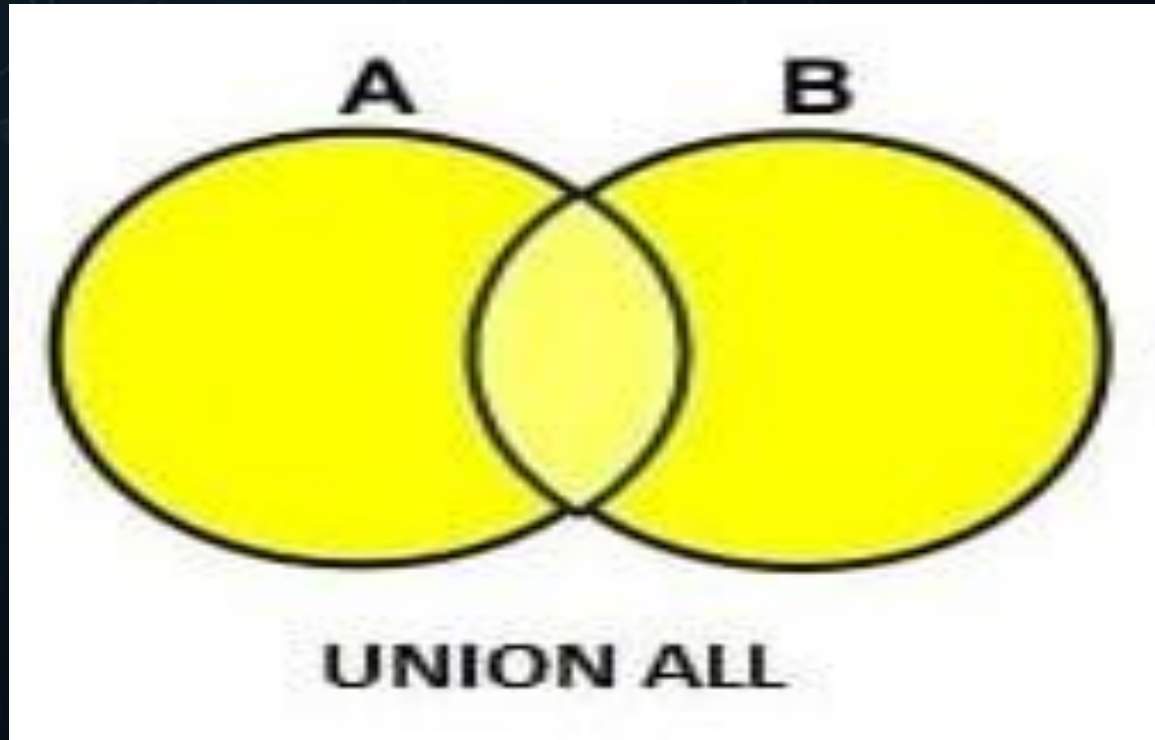
|_Consulta

Gerar **SELECTS** utilizando **INNER JOIN**, **LEFT JOIN**, **RIGHT JOIN** e **FULL JOIN**.

|_UNIONS



|_UNIONS



|_Consulta

Gerar **SELECTS** utilizando **UNION** e
UNION ALL.

| _Stored Procedures (Functions)

Vamos criar uma **FUNCTION**.

|_Views

Vamos criar uma **VIEW**.

|_Index


Vamos criar **INDEXs**

L >
< C

Obrigada(o)!

LET'S CODE

Av. Faria Lima, 1306
4º andar

(11) 2609-3807 
contato@letscode.com.br 