

# Introdução a Banco de Dados Relacionais

**Pâmela Apolinário Borges**

Engenheira de Software

**@pamelaapborges - LinkedIn**

# Objetivo Geral

O objetivo geral do curso é fornecer uma introdução aos Bancos de Dados Relacionais e desenvolver habilidades na criação, modelagem e consulta desses bancos

# Pré-requisitos

- ✓ Computador com acesso a internet
- ✓ Vontade de aprender

# Conteúdo Programático

- ❑ **Introdução aos Bancos de Dados Relacionais e conceitos básicos de SQL.**
- ❑ **Modelagem de tabelas, colunas e registros com operações CRUD.**
- ❑ **Chaves primárias e estrangeiras com modelagem de tabelas relacionadas.**

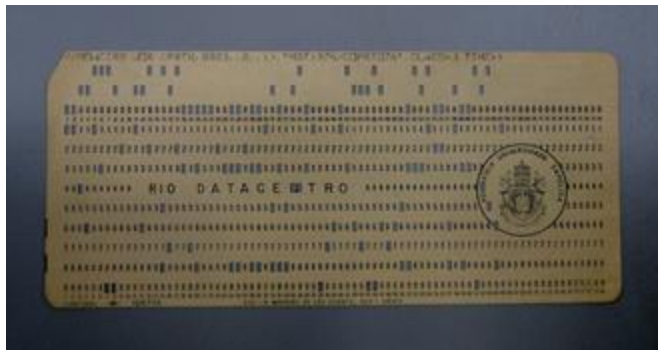
# Conteúdo Programático

- ☐ Normalização de dados, identificando e corrigindo problemas de normalização.
- ☐ Consultas avançadas com junções e subconsultas.
- ☐ Funções agregadas e agrupamento de resultados com GROUP BY e HAVING.
- ☐ Uso de índices para otimização de consultas.

# Conceitos Básico e Estrutura do Banco de dados Relacional

Introdução aos Bancos de Dados Relacionais

# O que é um Banco de Dados?



# Tipos de Banco de Dados

- ❑ Relacionais/SQL
- ❑ Não Relacionais/NoSQL (Not OnlySQL)
- ❑ Orientado a Objetos
- ❑ Hierárquico



# SGBD



# SGBD

Funcionalidades básicas:

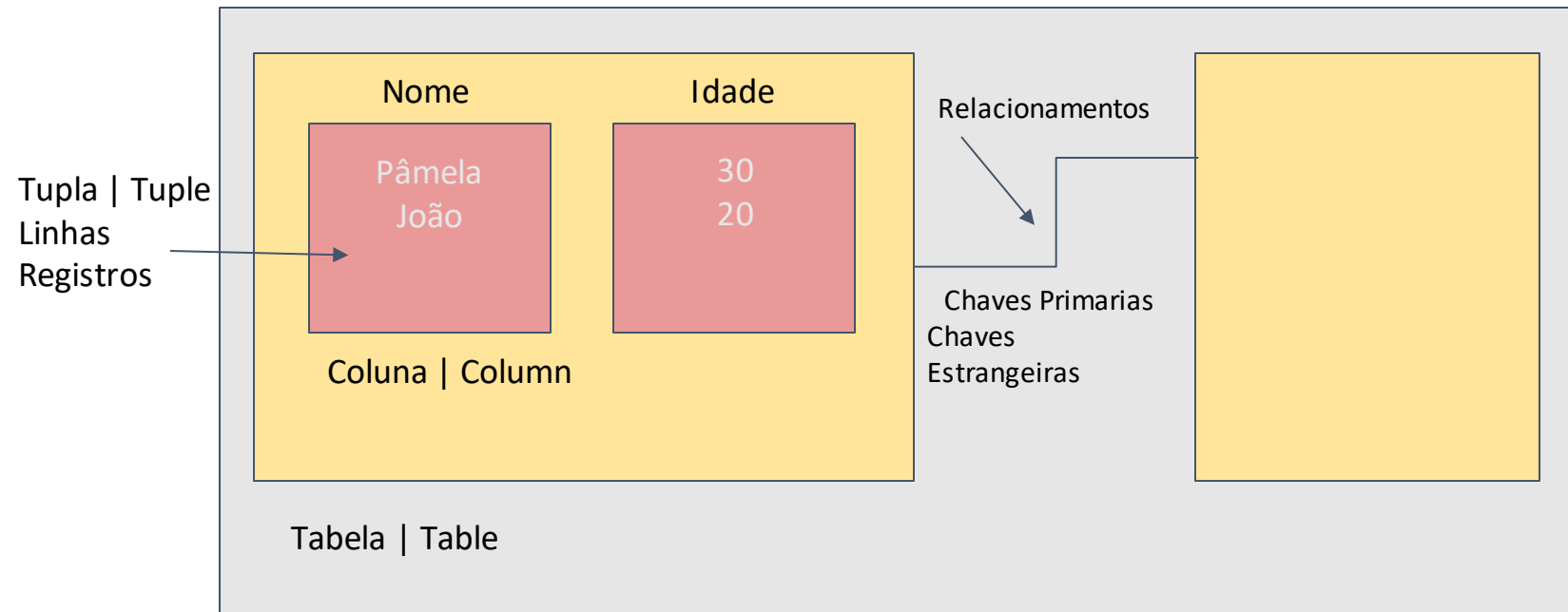
**C**reate

**R**ead

**U**pdate

**D**elelete

# Estrutura de um BD Relacional



# Características

- ❑ Relacionamento entre tabelas
- ❑ Linguagem de Consulta Estruturada (SQL)
- ❑ Integridade referencial
- ❑ Normalização de dados
- ❑ Segurança
- ❑ Flexibilidade e extensibilidade
- ❑ Suporte a transações ACID

# ACID

**A**tomicidade

**C**onsistência

**I**solamento

**D**urabilidade

# Links Úteis

- **Referências:**

- <https://www.oracle.com/br/database/what-is-a-relational-database/>

# Introdução e Conceitos Básicos de SQL

Introdução aos Bancos de Dados Relacionais

# SQL - Structured Query Language





# Organização da SQL

- ❑ DQL - Linguagem de Consulta de Dados
  - ❑ SELECT;
- ❑ DML - Linguagem de Manipulação de Dados
  - ❑ INSERT, UPDATE e DELETE;
- ❑ DDL - Linguagem de Definição de Dados
  - ❑ CREATE, ALTER, DROP;

# Organização da SQL

- ❑ DCL - Linguagem de Controle de Dados
  - ❑ GRANT, REVOKE
- ❑ DTL - Linguagem de Transação de Dados
  - ❑ BEGIN, COMMIT, ROLLBACK

# Sintaxe Básica: Nomenclatura



- ❑ Os nomes devem começar com uma letra ou com um caractere de sublinhado (\_)
- ❑ \*\*\* Os nomes podem conter letras, números e caracteres de sublinhado (\_).
- ❑ Sensibilidade a maiúsculas e minúsculas

# Links Úteis

- **Referências:**
  - <https://www.sqltutorial.org/>

# MER e DER: Modelagem de Bancos de Dados

Introdução aos Bancos de Dados Relacionais

# MER e DER

- ❑ O Modelo Entidade-Relacionamento (MER) é representado através de diagramas chamados Diagramas Entidade-Relacionamento (DER).

<https://app.creately.com/>

# Entidades

As entidades são nomeadas com substantivos concretos ou abstratos que representem de forma clara sua função dentro do domínio.



# Atributos

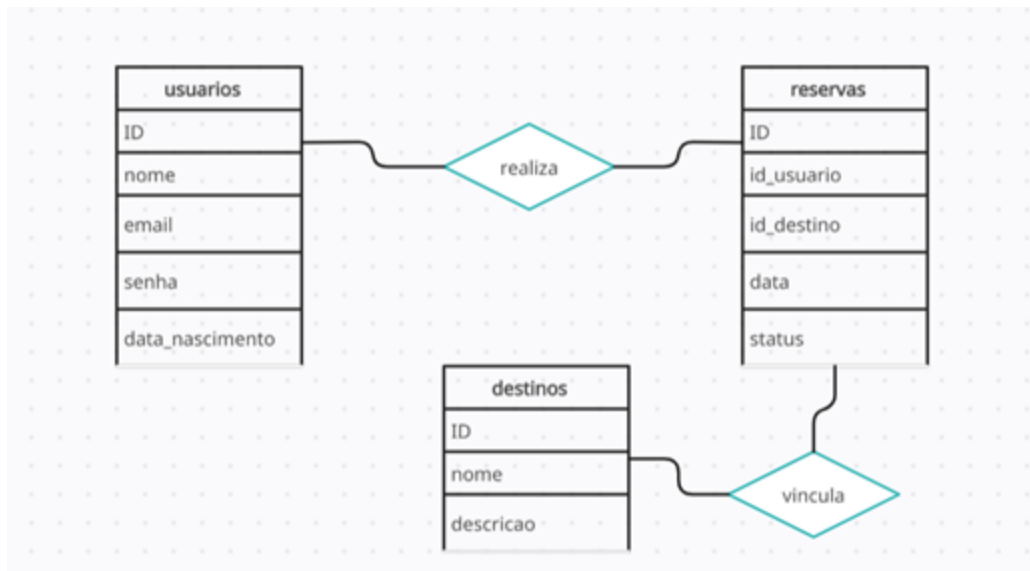
Os atributos são as características ou propriedades das entidades. Eles descrevem informações específicas sobre uma entidade.





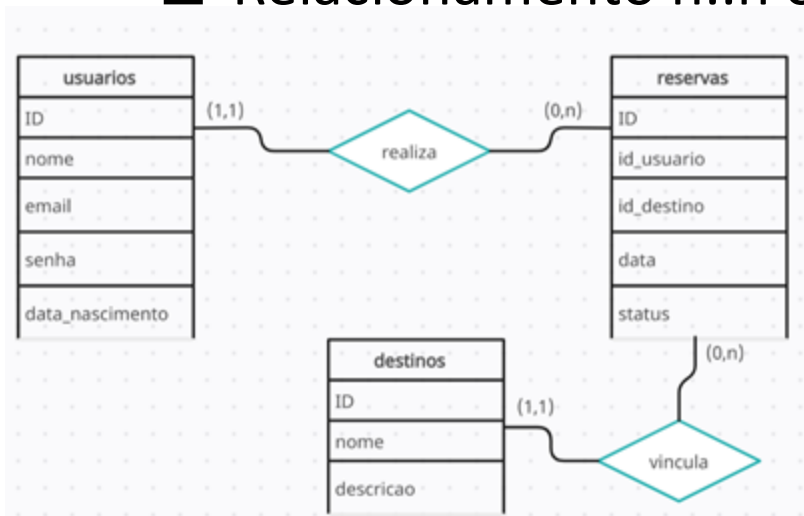
# Relacionamentos

Os relacionamentos representam as associações entre entidades



# Cardinalidade

- ❑ Relacionamento 1..1 (um para um)
- ❑ Relacionamento 1..n ou 1..\* (um para muitos)
- ❑ Relacionamento n..n ou \*..\* (muitos para muitos)



# Criando diagramas com IA

❏ <https://app.quickdatabasediagrams.com/>

# Links Úteis

- **Referências:**

- <https://www.lucidchart.com/pages/pt/o-que-e-diagrama-entidade-relacionamento>
- <https://app.creately.com/>

# Configuração do Ambiente

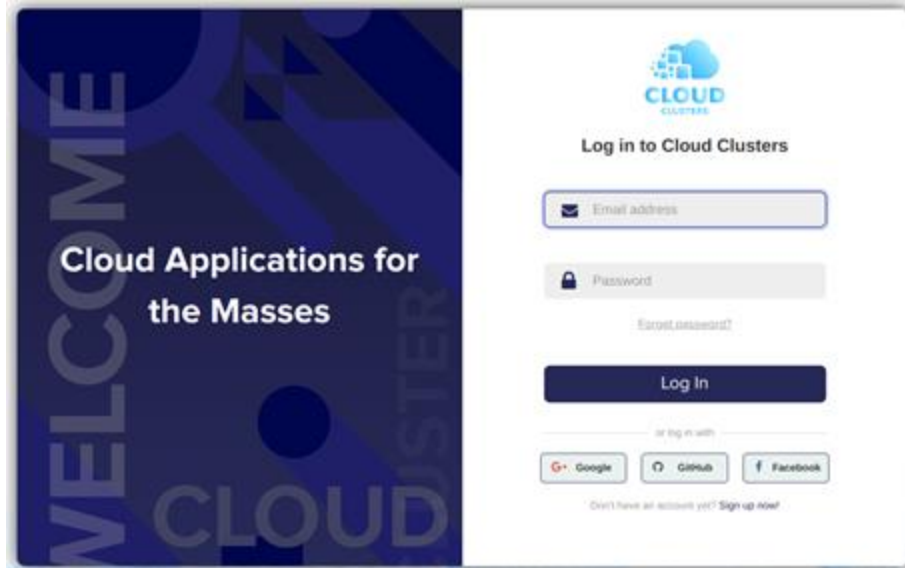
Introdução aos Bancos de Dados Relacionais

# Diagramas

❏ <https://app.creately.com/>

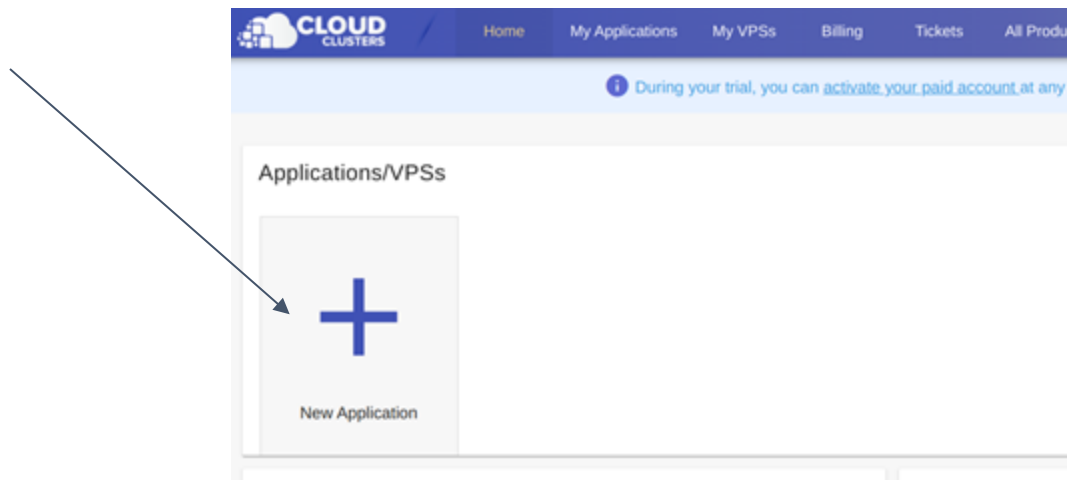
# Banco de Dados

1. Acessar a url: <https://clients.cloudclusters.io/> e criar conta.



# Banco de Dados

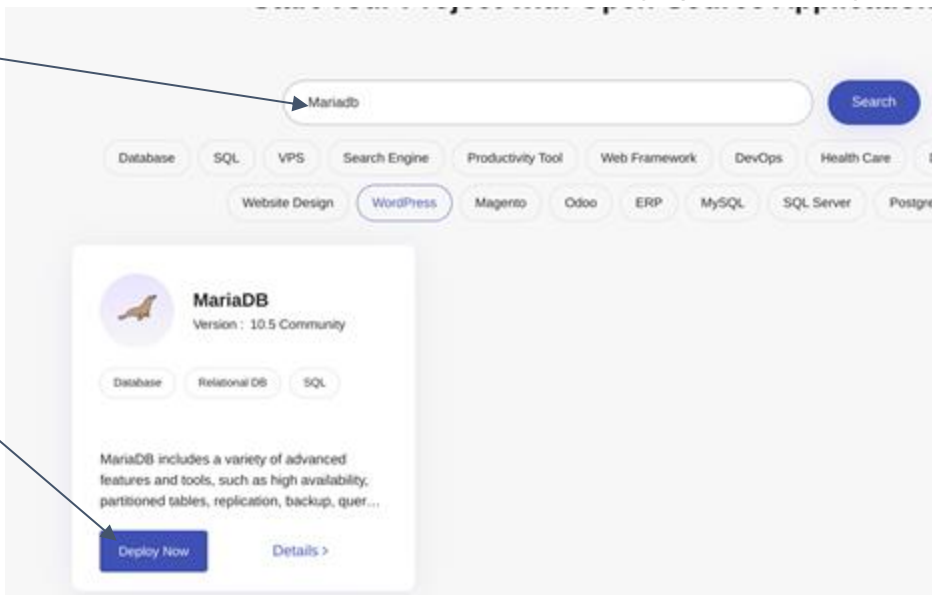
2. Selecionar opção “New Application”.





# Banco de Dados

3. Buscar MariaDB e selecionar a opção Deploy Now.



# Banco de Dados

## 4. Não alterar nada e selecionar a opção Free Trial Now

**MariaDB Cloud Hosting Free Trial**

✓ Try free for 7 days. Cancel anytime.  
✓ Enjoy all product's features and 24/7 free tech support.

**Plan**

☒ Express ☐ Basic ☐ Professional ☐ Advanced

CPU Cores:	2
Memory:	2GB
Disk:	60GB SSD
Backup Storage:	60GB SATA
Replica:	1

**Version**

☒ MariaDB@10.5 Community

**Component**

☒ MariaDB@10.5

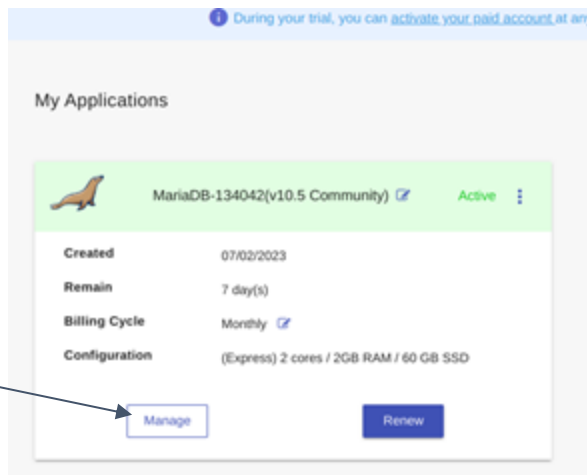
**Location**

☒ South U.S. ☐ Central U.S.

[Free Trial Now](#)

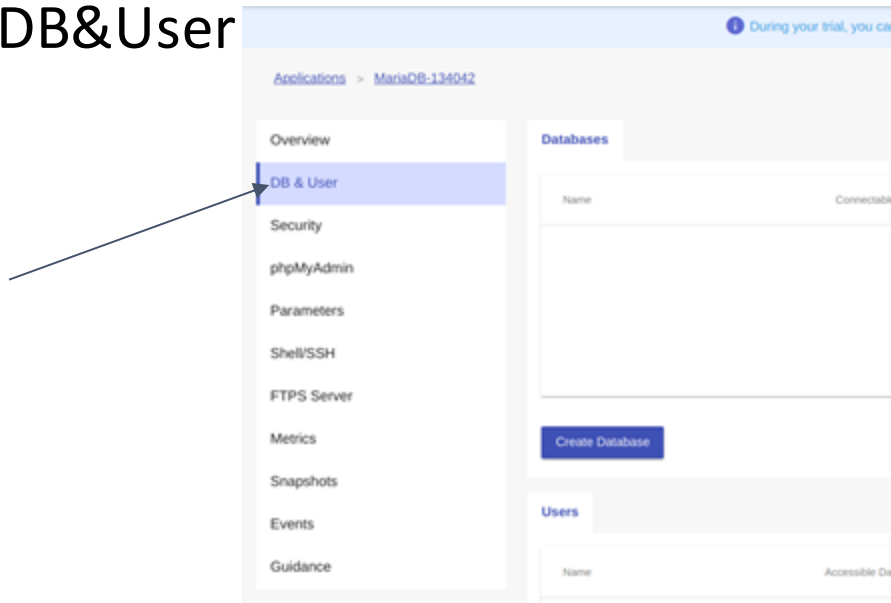
# Banco de Dados

5. Após finalizado o deploy será apresentado a opção “Manage”.



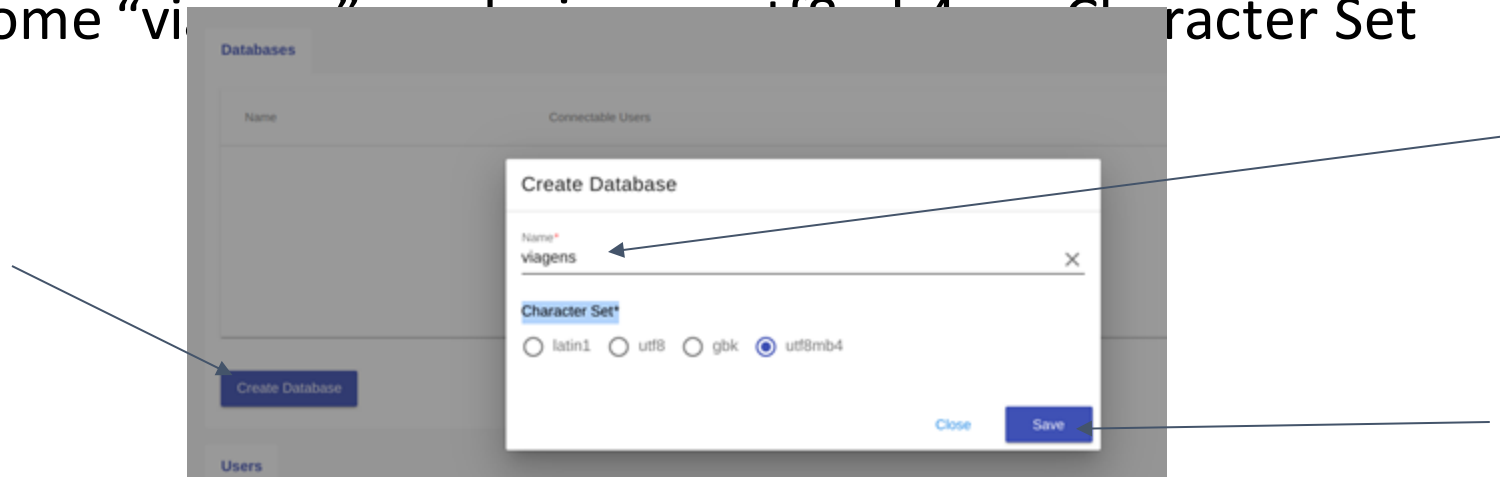
# Banco de Dados

6. Configurar Banco de Dados e Usuários, selecionando a opção DB&User



# Banco de Dados

7. Crie o Banco clicando em “Create Database” adicione o nome “viagens” e selecione o Character Set



# Banco de Dados

8. Crie um usuário clicando em “Create User”, adicionando os dados abaixo e a senha coloque uma de sua opção

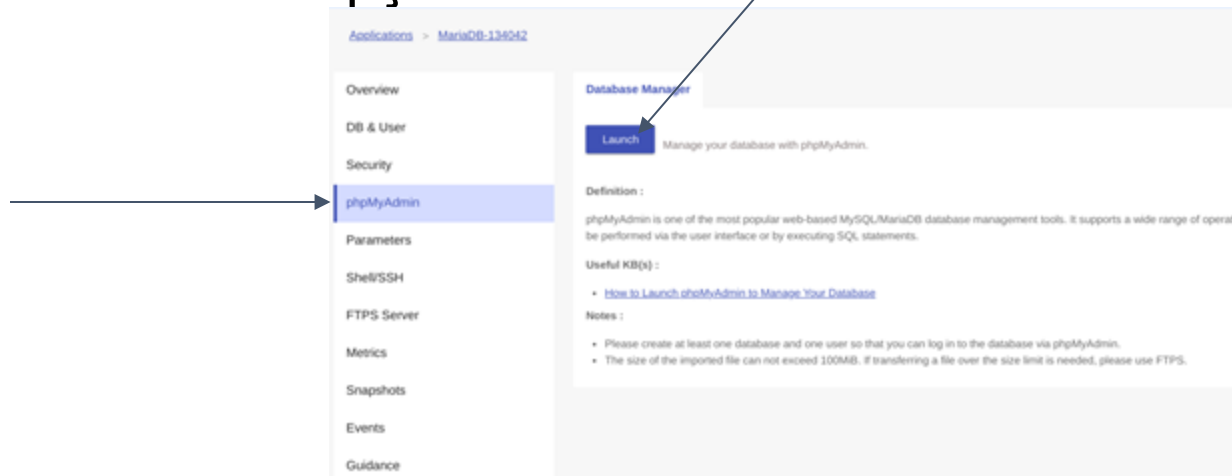
The image shows a 'New User' dialog box overlaid on a database management interface. The dialog box contains the following fields and elements:

- Name:** A text input field containing 'adm'.
- Permissions:** A dropdown menu showing 'Super'.
- Warning:** A message with a warning icon: 'Please do not change the default administrator password using super users, otherwise some application management features will be limited.'
- Password:** A text input field with masked characters (dots).
- Confirm Password:** A text input field with masked characters (dots).
- Buttons:** 'Close' and 'Save' buttons at the bottom right.

The background interface shows a 'Databases' section with a 'Create Database' button and a 'Users' section with a 'Create User' button.

# PhpMyAdmin

1. No menu esquerdo selecione a opção “phpMyAdmin” e seleciona a opção “Launch”



# PhpMyAdmin

2. Adicione as credenciais do usuário criado no passo 8 e selecione a opção “—”



The screenshot shows the phpMyAdmin login interface. At the top, there is a logo with a sailboat and the text "phpMyAdmin". Below the logo, it says "Bemvindo ao phpMyAdmin". There are two main sections: "Lingua - Language" and "Entrada". The "Lingua - Language" section has a dropdown menu currently showing "Português - Portuguese". The "Entrada" section has two input fields: "Utilizador :" and "Palavra-passe:". Below these fields is a button labeled "Executar".



# Links Úteis

- **Referências:**
  - <https://clients.cloudclusters.io/>

# Modelagem de Dados Relacionais - Tabelas, Colunas e Registros

Introdução aos Bancos de Dados Relacionais

# Tabelas

Ela é usada para armazenar dados de forma organizada. Cada tabela em um banco de dados relacional tem um nome único e é dividida em colunas e linhas

# Colunas

Uma coluna é uma estrutura dentro de uma tabela que representa um atributo específico dos dados armazenados. Cada coluna tem um nome único e um tipo de dados associado que define o tipo de informação que pode ser armazenado nela, como números, textos, datas, etc.

# Registros

Um registro, também conhecido como linha ou tupla, é uma instância individual de dados em uma tabela.

# Comando: CREATE TABLE

```
CREATE TABLE {{nome}}  
  
    ({{coluna}} {{tipo}} {{opções}} COMMENT  
    {{'COMENTARIO'}});
```

# Tipos de Dados

Os dados podem variar muito entre os diversos SGBD, os mais comuns são:

- ❑ Inteiro (Integer)
- ❑ Decimal/Numérico (Decimal/Numeric)
- ❑ Caractere/Varchar (Character/Varchar)
- ❑ Data/Hora (Date/Time)
- ❑ Booleano (Boolean)
- ❑ Texto longo (Text)

# Tipos de Dados

Os dados podem variar muito entre os diversos SGBD, os mais comuns são:

- ❑ Inteiro (Integer)
- ❑ Decimal/Numérico (Decimal/Numeric)
- ❑ Caractere/Varchar (Character/Varchar)
- ❑ Data/Hora (Date/Time)
- ❑ Booleano (Boolean)
- ❑ Texto longo (Text)



# Comando: CREATE TABLE -

## Opções



- ❑ Restrições de valor:
  - ❑ NOT NULL
  - ❑ UNIQUE
  - ❑ DEFAULT
- ❑ Chaves primárias e estrangeiras
- ❑ Auto Incremento

Hands On!

***“Falar é fácil.  
Mostre-me o código!”***

*Linus Torvalds*

# Comando: INSERT

INSERT INTO

{{ nome-tabela }}

([ coluna1, coluna2, ... ]) \*\*\* você pode ocultar as colunas

VALUES

([ valor-coluna1, valor-coluna2, ... ])

# Comando: SELECT

```
SELECT {{ lista_colunas }}
```

```
FROM tabela;
```

Onde \* retorna todas as colunas

# Comando: SELECT com Where

```
SELECT {{ lista_colunas }}
```

```
FROM tabela
```

```
WHERE {{ condicao }};
```

# Comando: SELECT - Operadores

- ☐ = (igualdade)
- ☐ <> ou != (desigualdade)
- ☐ > (maior que)
- ☐ < (menor que)
- ☐ >= (maior ou igual que)
- ☐ <= (menor ou igual que)
- ☐ LIKE (comparação de padrões)
- ☐ IN (pertence a uma lista de valores)
- ☐ BETWEEN (dentro de um intervalo)
- ☐ AND (e lógico)
- ☐ OR (ou lógico)

# Comandos: Update & Delete



# Comando: Update

UPDATE {{ tabela }}

SET

{{ coluna\_1 }} = {{ novo\_valor\_1 }},

{{ coluna\_2 }} = {{ novo\_valor\_2 }}

WHERE

{{ condicao }} ;



# Comando: Delete

DELETE FROM

{{ tabela }}

WHERE

{{ condicao }};

# Scripts produzidos no Hands On

❏ <https://github.com/pamelaborges/dio-bd-relacional>

# Links Úteis

- **Referências:**

- <https://mariadb.com/kb/en/data-types/>
- <https://mariadb.com/kb/en/create-table/>
- <https://clients.cloudclusters.io/>

# Modelagem de Dados Relacionais - Operações CRUD

Introdução aos Bancos de Dados Relacionais

# Comando: INSERT

INSERT INTO

{{ nome-tabela }}

([ coluna1, coluna2, ... ]) \*\*\* você pode ocultar as colunas

VALUES

([ valor-coluna1, valor-coluna2, ... ])

# Comando: SELECT

```
SELECT {{ lista_colunas }}
```

```
FROM tabela;
```

Onde \* retorna todas as colunas

# Comando: SELECT com Where

```
SELECT {{ lista_colunas }}
```

```
FROM tabela
```

```
WHERE {{ condicao }};
```

# Comando: SELECT - Operadores

- ☐ = (igualdade)
- ☐ <> ou != (desigualdade)
- ☐ > (maior que)
- ☐ < (menor que)
- ☐ >= (maior ou igual que)
- ☐ <= (menor ou igual que)
- ☐ LIKE (comparação de padrões)
- ☐ IN (pertence a uma lista de valores)
- ☐ BETWEEN (dentro de um intervalo)
- ☐ AND (e lógico)
- ☐ OR (ou lógico)



# Comandos: Update & Delete



# Comando: Update

UPDATE {{ tabela }}

SET

{{ coluna\_1 }} = {{ novo\_valor\_1 }},

{{ coluna\_2 }} = {{ novo\_valor\_2 }}

WHERE

{{ condicao }} ;

# Comando: Delete

DELETE FROM

{{ tabela }}

WHERE

{{ condicao }};

# Scripts produzidos no Hands On

❏ <https://github.com/pamelaborges/dio-bd-relacional>

# Links Úteis

- **Referências:**

- <https://mariadb.com/kb/en/data-types/>
- <https://mariadb.com/kb/en/create-table/>
- <https://clients.cloudclusters.io/>

# Modelagem de Dados Relacionais - Alterando e Excluindo Tabelas

Introdução aos Bancos de Dados Relacionais

# Problema:

Usuários com endereços longos não estão conseguindo realizar cadastro no sistema

Opções:

- Recriar a tabela, migrar os dados e excluir a tabela anterior
- Alterar estrutura da tabela

# Drop Table

O comando DROP TABLE é usado no SQL - para remover uma tabela existente de um banco de dados relacional.

Ele exclui permanentemente a tabela

```
DROP TABLE {{tabela}}
```



# Alter Table

A cláusula ALTER TABLE é usada no SQL para modificar a estrutura de uma tabela existente em um banco de dados relacional.

Ela permite:

- ☐ Adicionar, alterar ou excluir colunas
- ☐ Modificar as restrições, índices
- ☐ Renomear a tabela entre outras alterações

# Scripts produzidos no Hands On

❏ <https://github.com/pamelaborges/dio-bd-relacional>

# Links Úteis

- **Referências:**

- <https://mariadb.com/kb/en/alter-table/>
- <https://mariadb.com/kb/en/drop-table/>

# Modelagem de Dados Relacionais - Chaves Primária e Estrangeiras

Introdução aos Bancos de Dados Relacionais

# Chaves Primária

- ❑ Identifica exclusivamente
- ❑ Não pode conter valores nulos (NULL)
- ❑ Uma tabela pode ter apenas uma chave primária.

# Chaves Primária

```
CREATE TABLE {{tabela}}
```

```
( ID PRIMARY KEY AUTOINCREMENT,
```

```
... );
```



Constraint

```
ALTER TABLE {{tabela}}
```

```
MODIFY COLUMN ID INT PRIMARY KEY;
```

# Chaves Estrangeira

Ela é usada para estabelecer e manter a integridade dos dados entre tabelas relacionadas

- ❑ Pode ser nula (NOT NULL); \*\* registro órfão
- ❑ É possível ter mais de uma (ou nenhuma) em uma tabela.

# Chaves Estrangeira

```
CREATE TABLE {{tabela }} (  
    id INT PRIMARY KEY,  
    chave_estrangeira INT,  
    FOREIGN KEY (chave_estrangeira) REFERENCES {{outra  
tabela }} (id)  
);
```



# Chaves Estrangeira

**ALTER TABLE** {{ tabela }}

**ADD CONSTRAINT** {{nome\_constraint }}

**FOREIGN KEY** (ID\_)

**REFERENCES** {{outra\_tabela}} (ID)

# Chaves Estrangeira - Restrições

- ❑ ON DELETE especifica o que acontece com os registros dependentes quando um registro pai é excluído.
- ❑ ON UPDATE define o comportamento dos registros dependentes quando um registro pai é atualizado.
- ❑ CASCADE, SET NULL, SET DEFAULT e RESTRICT

Hands On!

***“Falar é fácil.  
Mostre-me o código!”***

*Linus Torvalds*

# Scripts produzidos no Hands On

❏ <https://github.com/pamelaborges/dio-bd-relacional>

# Links Úteis

- **Referências:**

- <https://mariadb.com/kb/en/data-types/>
- <https://mariadb.com/kb/en/create-table/>

# Normalização de Dados

Introdução aos Bancos de Dados Relacionais

# Problema:

id	nome	endereco
1	João	Rua A, 123, Cidade X, Estado Y
2	Maria	Rua B, 456, Cidade Y, Estado Z
3	Pedro	Avenida C, 789, Cidade X, Estado Y

Como buscar todos os usuário da Cidade X ?

# Normalização de dados

A normalização de dados é um processo no qual se organiza e estrutura um banco de dados relacional de forma a eliminar redundâncias e anomalias, garantindo a consistência e integridade dos dados.

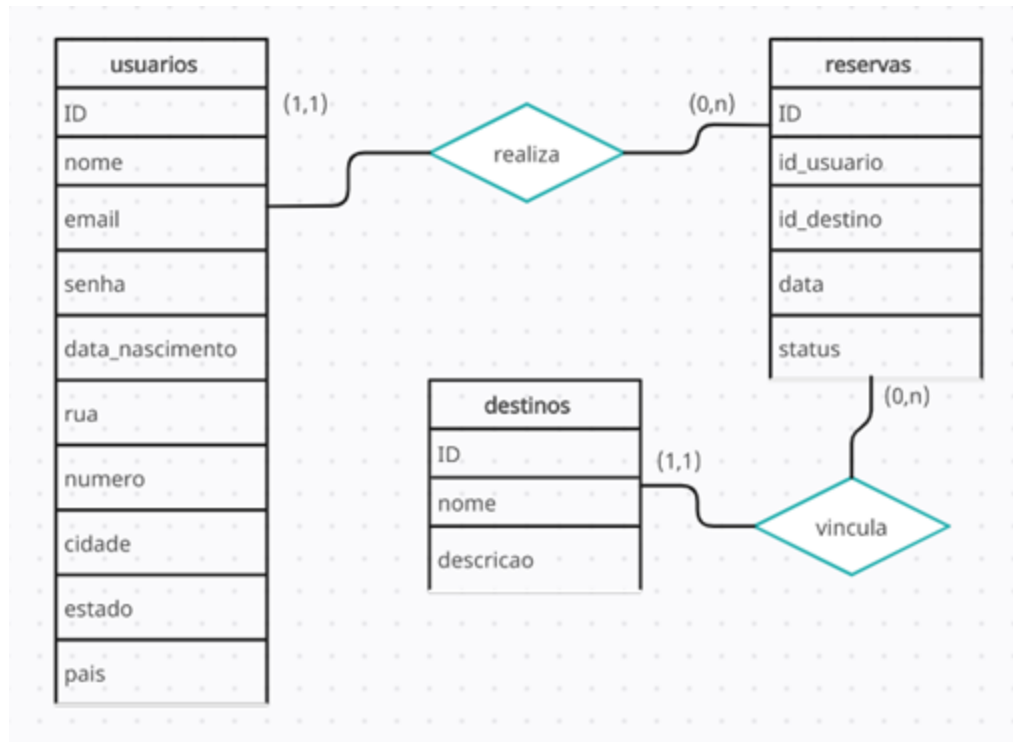


# Formas Normais

## 1FN: Atomicidade de dados

A 1FN estabelece que cada valor em uma tabela deve ser atômico, ou seja, indivisível. Nenhum campo deve conter múltiplos valores ou listas. No seu caso, o campo "endereco" contém múltiplos valores, como rua, número, cidade e estado. Para atingir a 1FN, precisamos dividir o campo "endereco" em colunas separadas.

# Formas Normais



# 2FN

- ❑ A 2FN estabelece que uma tabela deve estar na 1FN .
- ❑ Todos os atributos não chave devem depender totalmente da chave primária.

Dica se sua tabela tem uma **chave primária simples** não existe a possibilidade de termos dependência parcial e por tanto ela já se encontra na 2FN

# 3FN

- ❑ Uma tabela deve estar na 2FN .
- ❑ Nenhuma coluna não-chave depender de outra coluna não-chave.

Nosso exemplo: Relação Estado -> Cidade

# Resumo

- ❑ A 1FN garante que cada valor seja atômico e que os registros sejam únicos e identificáveis.
- ❑ A 2FN garante que os atributos não chave dependam totalmente da chave primária, evitando dependências parciais.
- ❑ A 3FN elimina dependências transitivas entre os atributos não chave, garantindo que cada atributo não chave dependa apenas da chave primária, não havendo dependências indiretas entre eles.

# Formas Normais

São 6 ao todo, para mais detalhes consultar

[https://pt.wikipedia.org/wiki/Normaliza%C3%A7%C3%A3o  
de dados](https://pt.wikipedia.org/wiki/Normaliza%C3%A7%C3%A3o_de_dados)

# Scripts produzidos no Hands On

❏ <https://github.com/pamelaborges/dio-bd-relacional>

# Links Úteis

- **Referências:**

- [https://pt.wikipedia.org/wiki/Normaliza%C3%A7%C3%A3o\\_de\\_dados](https://pt.wikipedia.org/wiki/Normaliza%C3%A7%C3%A3o_de_dados)



# Consultas Avançadas - Consultas com junções e subconsultas

Introdução aos Bancos de Dados Relacionais

# Junções: JOINS

São usadas no SQL para combinar dados de duas ou mais tabelas relacionadas em uma única consulta

# Junções: Tipos

- ❑ INNER JOIN
- ❑ LEFT JOIN ou LEFT OUTER JOIN
- ❑ RIGHT JOIN ou RIGHT OUTER JOIN
- ❑ FULL JOIN ou FULL OUTER JOIN

# INNER JOIN

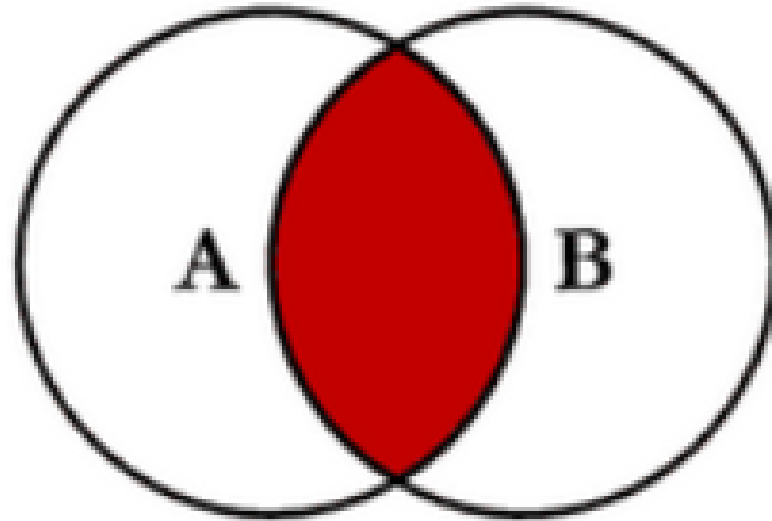
Retorna apenas as linhas que têm correspondência em ambas as tabelas envolvidas na junção. A junção é feita com base em uma condição de igualdade especificada na cláusula ON.

```
SELECT *
```

```
FROM tabela1
```

```
INNER JOIN tabela2 ON tabela1.coluna = tabela2.coluna;
```

# INNER JOIN



# LEFT JOIN

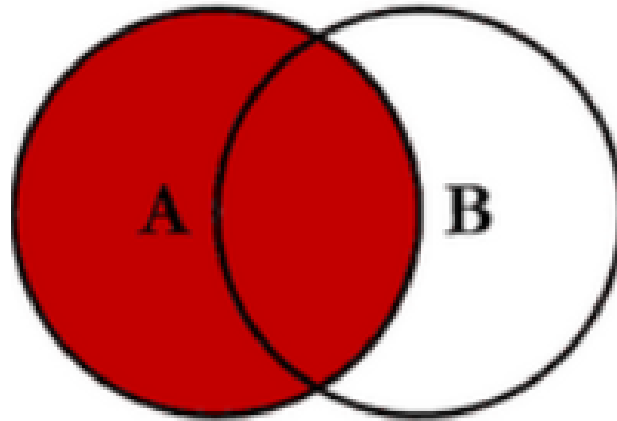
Retorna todas as linhas da tabela à esquerda da junção e as linhas correspondentes da tabela à direita. Se não houver correspondência, os valores da tabela à direita serão NULL.

```
SELECT *
```

```
FROM tabela1
```

```
LEFT JOIN tabela2 ON tabela1.coluna = tabela2.coluna;
```

# LEFT JOIN



# RIGHT JOIN

Retorna todas as linhas da tabela à direita da junção e as linhas correspondentes da tabela à esquerda. Se não houver correspondência, os valores da tabela à esquerda serão NULL.

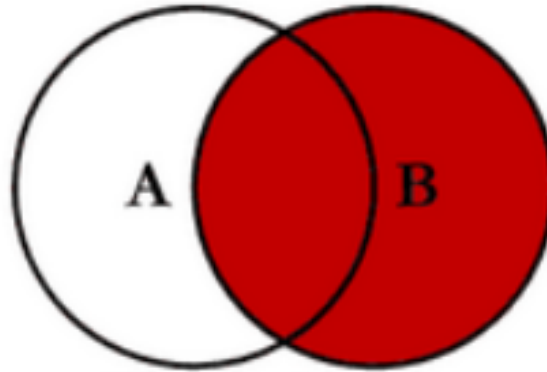
```
SELECT *
```

```
FROM tabela1
```

```
RIGHT JOIN tabela2 ON tabela1.coluna = tabela2.coluna;
```



# RIGHT JOIN



# FULL JOIN

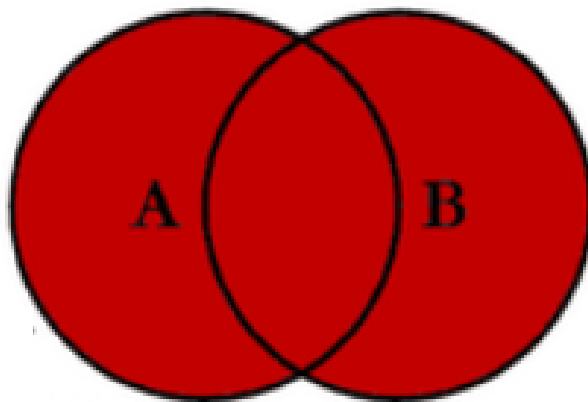
Retorna todas as linhas de ambas as tabelas envolvidas na junção, combinando-as com base em uma condição de igualdade. Se não houver correspondência, os valores ausentes serão preenchidos com NULL.

```
SELECT *
```

```
FROM tabela1
```

```
FULL JOIN tabela2 ON tabela1.coluna = tabela2.coluna;
```

# FULL JOIN



MariaDB não tem comando full join

# Sub Consultas

Elas permitem realizar consultas mais complexas permitindo que você use o resultado de uma consulta como entrada para outra consulta.

# Sub Consultas

As subconsultas podem ser usadas em várias partes de uma consulta:

- ☐ SELECT
- ☐ FROM
- ☐ WHERE
- ☐ HAVING
- ☐ JOIN.

# Scripts produzidos no Hands On

❏ <https://github.com/pamelaborges/dio-bd-relacional>

# Links Úteis

- **Referências:**
  - <https://mariadb.com/kb/en/joins/>

# Consultas Avançadas - Funções agregadas e Agrupamento de resultados

Introdução aos Bancos de Dados Relacionais



# Funções Agregadas

- ❑ COUNT: Conta o número de registros.
- ❑ SUM: Soma os valores de uma coluna numérica.
- ❑ AVG: Calcula a média dos valores de uma coluna numérica.
- ❑ MIN: Retorna o valor mínimo de uma coluna.
- ❑ MAX: Retorna o valor máximo de uma coluna.

# Agrupamento de Resultados

SELECT ...

FROM ...

GROUP BY

# Limite de Resultados

SELECT ...

FROM ...

GROUP BY ...

LIMIT {{numero}}

OFFSET {{numero}} \*\*\* opcional

# Ordenação de Resultados

SELECT ...

FROM ...

ORDER BY

# Ordenação de Resultados

- ❑ ASC
- ❑ DESC
- ❑ Múltiplas Colunas

# Scripts produzidos no Hands On

❏ <https://github.com/pamelaborges/dio-bd-relacional>

# Links Úteis

- **Referências:**

- <https://mariadb.com/kb/en/aggregate-functions/>

# Consultas Avançadas - Índices de Busca

Introdução aos Bancos de Dados Relacionais



# Análise do Plano de Execução

Ela nos permite examinar as operações realizadas, as tabelas acessadas, os índices utilizados e outras informações importantes para identificar possíveis melhorias de desempenho.

# Análise do Plano de Execução

EXPLAIN

SELECT \*

FROM {{TABELA}}

....

# Análise do Plano de Execução

- ❑ `select_type`: "SIMPLE", "SUBQUERY", "JOIN"
- ❑ `table`.
- ❑ `type`: "ALL", "INDEX" entre outros
- ❑ `possible_keys`: Os índices possíveis que podem ser utilizados na operação.
- ❑ `key`: O índice utilizado na operação, se aplicável.
- ❑ `key_len`: O comprimento do índice utilizado.
- ❑ `ref`: As colunas ou constantes usadas para acessar o índice.
- ❑ `rows`

# Índices de Busca

Esses recursos são fundamentais para melhorar o desempenho das consultas e otimizar a recuperação de informações em bancos de dados.

# Índices de Busca

```
CREATE INDEX {{nome_index}}  
ON {{tabela}} ({{coluna1, coluna2...}});
```

# Scripts produzidos no Hands On

❏ <https://github.com/pamelaborges/dio-bd-relacional>

# Links Úteis

- **Referências:**

- <https://mariadb.com/kb/en/alter-table/#add-index>

# Dúvidas?

> Fórum/Artigos - <https://web.dio.me/articles>