Competências, Habilidades e Bases Tecnológicas da disciplina de Banco de Dados II

II.3 BANCO DE DADOS II					
Função: Manipulação de dados em bancos de dados relacionais **Classificação:** Execução					
Atribuições e Responsabilidades					
Manipular dados, utilizando sistemas gerenciadores de bancos de dados relacionais.					
Valores e Atitudes					
Incentivar a criatividade. Desenvolver a criticidade. Estimular o interesse na resolução de situações-problema.					
Competência			**Habilidades**		
1. Efetuar operações em bancos de dados com SQL.			1.1 Executar comandos SQL para manipulação de dados. 1.2 Utilizar transações para garantia de integridade em bancos de dados.		
Bases Tecnológicas					
Práticas de SQL com SGBDR • Linguagem de manipulação de dados (DML); • Linguagem de consulta de dados (DQL); • Transações. DML • Inserção; • Atualização; • Exclusão. DQL • Projeção, seleção, renomeação; • Ordenação; • Agrupamento e funções agregadas; • Junção interna; • Junções externas à esquerda e à direita; • Produto cartesiano (*full/cross join*); • União, interseção e diferença. Transações • Operações *ACID;* • *COMMIT* e *ROLLBACK.*					
Carga horária (horas-aula)					
Teórica	**00**	**Prática Profissional**	**60**	**Total**	**60 Horas-aula**
Teórica (2,5)	**00**	**Prática Profissional (2,5)**	**50**	**Total (2,5)**	**00 Horas-aula**
Possibilidade de divisão de classes em turmas, conforme o item 4.8 do Plano de Curso.					

Grupo de Formulação e Análises Curriculares - Centro Paula Souza / SP

Observações a considerar:

1-) Comunicação de alunos com alunos e professores:

- Microsoft Teams;
- Criação de grupo nas redes sociais também é interessante.

2-) Uso de celulares:

Para o bom andamento das aulas, recomendo que utilizem os celulares em *vibracall*, para não atrapalhar o andamento da aula.

3-) Material das aulas:

A disciplina trabalha com **NOTAS DE AULA** que são disponibilizadas ao final de cada aula no Github de forma pública em: https://github.com/marcossousa33dev/aulaBDII202502

4-) Prazos de trabalhos e atividades:

Toda atividade solicitada terá uma data limite de entrega, de forma alguma tal data será postergada ou seja, se não for entregue até a data limite a mesma receberá menção I, isso tanto para atividades entregues de forma impressa, digital ou no Microsoft Teams.

5-) Qualidade do material de atividades:

- Impressas ou manuscritas:
  Muita atenção na qualidade do que será entregue, atividades sem grampear, faltando nome e número de componentes, rasgadas, amassadas, com rebarba de folha de caderno e etc. serão desconsiderados por mim.

- Digitais:
  Ao enviarem atividades para o e-mail, ou inseridas no Microsoft Teams, **SEMPRE** deverá ter o nome da atividade que está sendo enviada, e no corpo do e-mail ou da atividade deverá ter o(s) nome(s) do(s) integrante(s), sem estar desta forma a atividade será **DESCONSIDERADA**.

6-) Menções e critérios de avaliação:

Na ETEC os senhores serão avaliados por MENÇÃO, onde temos:
- MB - Muito Bom;
- B - Bom;
- R - Regular;
- I - Insatisfatório.

Cada trimestres poderemos ter as seguintes formas de avaliação:
- Avaliação teórica;
- Avaliação prática (a partir do 2º trimestre, e as turmas do 2º módulo em diante);
- Seminários;
- Trabalhos teóricos e/ou práticos;
- Assiduidade;
- Outras que se fizerem necessário.

Caso o aluno tenha alguma menção I em algum dos trimestres será aplicada uma recuperação, que poderá ser em forma de trabalho prático ou teórico.

1. **Recomendações bibliográficas:**

CASTRO, Eduardo. Modelagem Lógica de Dados: construção básica e simplificada. São Paulo. Ciência Moderna, Agosto 2012.

MARTELLI, Richard; FILHO, Ozeas; CABRAL, Alex. Modelagem e banco de dados. São Paulo. Senac, 2017.

PEREIRA, Paloma. Introdução a banco de dados. São Paulo. Senac, 2020.

NACIONAL, Senac. Modelagem de dados. Rio de Janeiro. Senac, 2014.

2. **Revisão SQL**

2.1. O MySQLWorkBench.

No decorrer desse curso veremos uma série de recursos de gerenciamento ligados ao Sistema Gerenciador de Banco de Dados Relacionais (**SGBDR**) "**Mysql**", esses recursos visam desde a manutenção dos bancos de dados e suas tabelas através de recursos como "**triggers**", "***stored procedures***", "***views***", "***index***" recursos esses fundamentais para a boa "**saúde**" de um banco de dados no que se refere a consistência dos dados e performance, passando também pela parte de segurança referente a criação e gestão de usuários e atribuição e revogação de direitos a esses usuários além de formas de realizar backups das bases de dados existentes.

Tais tarefas podem se mostrar desafiadoras ao "***Database Administrator***" (**DBA**) profissional responsável por tais tarefas, logo se faz necessário o uso de uma ou mais ferramentas que permita um alto grau de produção para esse profissional. Existe uma gama enorme de ferramentas capazes de fornecer esse auxilio para os mais diversos SGBDR corporativos existentes hoje no mercado. No nosso caso estamos trabalhando com o "**MySQL**" e vamos utilizar uma "**Integrated Development Envriroment**" (**IDE**) ou "**Ambiente Integrado de Desenvolvimento**" conhecido como "**MySQL Workbench**".

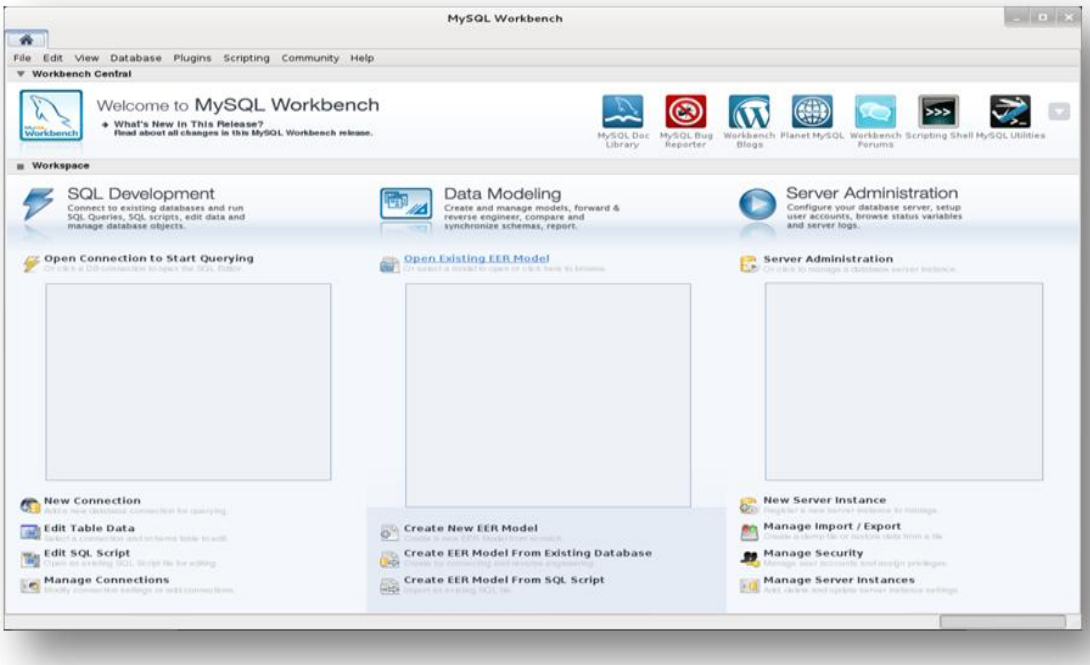
Essa ferramenta reúne dentro de si recursos divididos em três grupos básicos são eles:

- **SQL Development –** Reúne ferramentas ligadas a codificação de comandos **SQL Data Manipulation Language ou Linguagem de Manipulação de Dados** (**DML**), **Data definition Language** ou **Linguagem de Definição de Dados** (**DDL**), **Data Control Language** ou **Linguagem de controle de Dados** (**DCL**), **Data Transaction Language** ou **Linguagem de Transação de Dados** (**DTL**) e **Data Query Language ou Linguagem de Consulta de dados** (**DQL**).

- **Data Modeling –** Reúne ferramentas ligadas a modelagem de banco de dados permitindo a criação de diagramas/modelos de entidade relacionamento, além disso é possível converter esses diagramas em projetos físicos (bancos de dados) através de interface gráfica e podemos também realizar processos de engenharia reversa onde podemos "apontar" para um projeto físico de um determinado banco de dados e converter esse em diagramas/modelos de entidade relacionamento isso é muito útil principalmente quando estamos trabalhando em projetos que foram legados (já existiam e nos foram passados a título de continuação) e por alguma razão qualquer a equipe anterior não documentou a base de dados ou até mesmo essa documentação foi perdida.

- **Server Administrator –** Reúne ferramentas uteis para a gerencia de backups e da "saúde" das bases de dados criadas na instancia do "**MySQL**" que se está gerenciando, permite por exemplo verificar quantas conexões estão ativas nas bases de dados existentes, verificar a "saúde" da memória principal do servidor que hospeda a instancia do "**MySQL**", o trafego de dados, índice de eficiência das chaves, criar e associar a usuários diretos ou remover usuários e direitos, realizar backups entre outras ações.

A imagem abaixo ilustra a tela inicial do "**MySQL WorkBench**" onde podemos escolher entre os grupos de ferramentas citadas:



2.2. Aprimorando os comandos vistos

Os comandos da linguagem SQL são subdivididos em algumas categorias de comandos como: DDL, DML e DCL.

Nesta primeira etapa de BDII, usaremos a categoria DDL (*Data Definition Language* – Linguagem de Definição de Dados). Os comandos DDL são usados para definir a estrutura do banco de dados, organizando em tabelas que são compostas por campos (colunas). Comandos que compõem a DDL: CREATE, ALTER, DROP.

2.3. Tipos de Dados

Nas linguagens de programação, há quatro tipos primitivos de dados que são: inteiro, real, literal (texto) e lógico. Em cada SGBD, existem tipos de dados derivados desses tipos primitivos. No SGBD MySQL, é possível destacar os seguintes:

Categoria	Tipo de Dado	Descrição
Numéricos Inteiros	TINYINT (1 byte)	Inteiros de -128 a 127 (ou 0 a 255 sem sinal)
	SMALLINT (2 bytes)	Inteiros de -32.768 a 32.767
	MEDIUMINT (3 bytes)	Inteiros de -8.388.608 a 8.388.607
	INT ou INTEGER (4 bytes)	Inteiros de -2.147.483.648 a 2.147.483.647
	BIGINT (8 bytes)	Inteiros de -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
Numéricos Decimais	DECIMAL(M,D) ou NUMERIC	Números exatos com precisão definida (ex: DECIMAL(10,2))
	FLOAT(p)	Ponto flutuante de precisão simples
	DOUBLE ou DOUBLE PRECISION	Ponto flutuante de precisão dupla
Texto	CHAR(n)	Texto de tamanho fixo (até 255 caracteres)
	VARCHAR(n)	Texto de tamanho variável (até 65.535 bytes, dependendo do charset)
	TEXT	Texto longo (até 65.535 caracteres)
	TINYTEXT, MEDIUMTEXT, LONGTEXT	Variantes para textos maiores
Data e Hora	DATE	Data no formato AAAA-MM-DD
	DATETIME	Data e hora no formato AAAA-MM-DD HH:MM:SS
	TIMESTAMP	Data e hora com fuso horário
	TIME	Hora no formato HH:MM:SS
	YEAR	Ano no formato AAAA
Lógicos	BOOLEAN ou BOOL	Alias para TINYINT(1) (0 = falso, 1 = verdadeiro)
Outros	ENUM('a','b',...)	Lista de valores possíveis

No MySQL, os campos ainda possuem atributos que definem a validação dos valores, tais como:

Tipo de Validação	Comando / Sintaxe	Descrição
Obrigatoriedade	NOT NULL	Garante que o campo não aceite valores nulos.
Unicidade	UNIQUE	Garante que os valores da coluna sejam únicos.
Chave Primária	PRIMARY KEY	Combina NOT NULL e UNIQUE.
Intervalo de valores	CHECK (coluna condição)	Restringe os valores com base em uma condição lógica.

2.4 Comandos de DDL

DDL (Data Definition Language) é um subconjunto da linguagem SQL responsável por **definir, criar, modificar e excluir estruturas de dados** em um banco de dados, como **tabelas, índices, esquemas, visões e restrições**.

Em outras palavras, DDL trata da **estrutura** do banco de dados, e não dos dados em si.

Os principais comandos são:

Comando	Descrição
CREATE	Cria novos objetos no banco de dados (ex: tabelas, visões, índices).
ALTER	Modifica a estrutura de objetos existentes (ex: adicionar ou remover colunas).
DROP	Exclui objetos do banco de dados.
TRUNCATE	Remove todos os dados de uma tabela, mas mantém sua estrutura.
RENAME	Renomeia objetos do banco de dados.

Características Importantes:

- Comandos DDL geralmente **não podem ser revertidos** com ROLLBACK, pois são **auto-commit** (executam e salvam automaticamente).
- São essenciais na **modelagem e manutenção da estrutura** do banco de dados.
- Devem ser usados com cuidado, especialmente DROP e TRUNCATE, pois podem causar perda de dados.

A seguir iremos ver cada um deles em detalhes.

2.4.1 Criar Banco de Dados (CREATE DATABASE)

O comando CREATE DATABASE é usado para **criar um novo banco de dados** no MySQL. Ele define um espaço lógico onde você poderá criar tabelas, visões, procedimentos, etc.

Sintaxe:

```
CREATE DATABASE loja;
```

2.4.2 Apagar Banco de Dados (DROP DATABASE)

O comando DROP DATABASE é usado para **excluir permanentemente** um banco de dados e todos os seus objetos (tabelas, dados, procedimentos, etc.).

Sintaxe:

```
DROP DATABASE loja;
```

2.4.3. Criar tabela (CREATE TABLE)

O comando CREATE TABLE é usado para criar uma nova tabela em um banco de dados MySQL. Ele define o nome da tabela, as colunas, os tipos de dados de cada coluna e as restrições (como chaves primárias, valores obrigatórios, etc.).

Sintaxe:

```
CREATE TABLE nome_da_tabela (
   nome_coluna1 tipo_dado [restrições],
   nome_coluna2 tipo_dado [restrições],
   ...
   [restrições_de_tabela]
);
```

- nome_da_tabela: nome da nova tabela.
- nome_coluna: nome de cada coluna.
- tipo_dado: tipo de dado (ex: INT, VARCHAR, DATE, etc.).
- restrições: como NOT NULL, UNIQUE, PRIMARY KEY, DEFAULT, CHECK, etc.

Exemplo:

```sql
#CRIAÇÃO DE TABELA
CREATE TABLE clientes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(150) UNIQUE,
    data_nascimento DATE,
    genero ENUM('Masculino', 'Feminino', 'Outro'),
    ativo BOOLEAN DEFAULT TRUE,
    CHECK (year(data_nascimento) < 2025)
);
```

Explicação:

- id: chave primária com incremento automático.
- nome: obrigatório (NOT NULL).
- email: deve ser único.
- data_nascimento: aceita apenas datas.
- genero: limitado a três opções.
- ativo: valor padrão é TRUE.
- CHECK: garante que o cliente tenha nascido antes de 2025.

Você sabe o que é uma chave primária composta?

Definir uma **chave primária composta** no MySQL, você precisa especificar **duas ou mais colunas** que juntas identificam de forma única cada registro da tabela.

Vamos adaptar o exemplo anterior da tabela clientes para usar uma chave primária composta. Suponha que queremos que a combinação de email e data_nascimento seja única e sirva como identificador principal.

```sql
CREATE TABLE clientes (
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(150) NOT NULL,
    data_nascimento DATE NOT NULL,
    genero ENUM('Masculino', 'Feminino', 'Outro'),
    ativo BOOLEAN DEFAULT TRUE,
    criado_em DATETIME DEFAULT CURRENT_TIMESTAMP,
    CHECK (year(data_nascimento) < 2025),
    PRIMARY KEY (email, data_nascimento)
);
```

2.4.4. Apagar tabela (DROP TABLE)

```sql
DROP TABLE clientes;
```

2.4.5. Zerar dados de uma tabela (TRUNCATE TABLE)

O comando TRUNCATE TABLE no MySQL é usado para remover rapidamente todos os registros de uma tabela, sem registrar a exclusão linha por linha no log de transações (como acontece com o DELETE).

Características principais

- Remove todos os dados da tabela, mas mantém a estrutura (colunas, índices, etc.).
- Mais rápido e consome menos recursos do que DELETE FROM tabela sem WHERE.
- Reseta auto_increment (se houver uma coluna AUTO_INCREMENT).

2.4.6. ALTER TABLE

Para não se apagar uma tabela e recriá-la, é possível fazer alterações em sua estrutura por meio do comando ALTER TABLE. Isso é importante pois a execução do comando ***DROP TABLE*** apaga (obviamente) todos os registros da tabela, já a execução do comando ***ALTER TABLE*** não exclui nenhum registro.

- **Adicionar um campo**

Sintaxe:
ALTER TABLE nome_da_tabela
ADD nome_coluna tipo_dado [restrições];

Exemplo:

```
#ALTER TABLE adicionando campo
ALTER TABLE clientes
ADD telefone VARCHAR(20) DEFAULT 'Não informado';
```

- **Adicionar um campo posicionando depois de um campo já existente**

Sintaxe:
ALTER TABLE nome_da_tabela ADD nome_do_campo tipo_de_dado atributos AFTER nome_do_campo_ja_existente

Exemplo:

```
ALTER TABLE clientes
ADD endereco varchar(90) NOT NULL AFTER nome;
```

- **Alterar o tipo de dado ou tamanho de um campo**

Sintaxe:
ALTER TABLE nome_da_tabela MODIFY nome_do_campo tipo_de_dado

Exemplo:

```
ALTER TABLE clientes
MODIFY endereco VARCHAR(120);
```

- **Renomear um campo e modificar o tipo**

Sintaxe:
ALTER TABLE nome_da_tabela CHANGE COLUMN nome_do_campo novo_nome tipo atributos;

Exemplo para mudar apenas o nome (o tipo do campo é mantido):

```
ALTER TABLE clientes
CHANGE COLUMN data_nascimento datanasc date;
```

Exemplo para mudar o nome e o tipo do campo:

```
ALTER TABLE clientes
CHANGE COLUMN data_nascimento datahoranasc datetime;
```

- **Renomear uma tabela**

Sintaxe:
ALTER TABLE nome_da_tabela RENAME TO novo_nome_da_tabela

Exemplo:
```
ALTER TABLE clientes RENAME TO pessoas_fisicas;
```

- **Apagar um campo**

Sintaxe:
ALTER TABLE nome_da_tabela DROP COLUMN nome_do_campo

Exemplo:
```
ALTER TABLE clientes DROP COLUMN endereco;
```

- **Adicionar uma PRIMARY KEY**

Sintaxe:
ALTER TABLE nome_da_tabela ADD PRIMARY KEY(campo1,...)

Exemplo:
```
ALTER TABLE clientes
ADD PRIMARY KEY (cpf);
```

- **Apagar uma PRIMARY KEY**

Sintaxe:
ALTER TABLE nome_da_tabela DROP PRIMARY KEY

Exemplo:
```
ALTER TABLE clientes
DROP PRIMARY KEY;
```

2.5. FOREIGN KEY (Chave estrangeira)

Uma **chave estrangeira** é uma restrição que **cria um vínculo entre duas tabelas**, garantindo que os valores de uma coluna (ou conjunto de colunas) em uma tabela correspondam aos valores da **chave primária** (ou UNIQUE) de outra tabela. Ela é usada para **manter a integridade referencial** entre os dados.

Exemplo:

1. Criar a tabela clientes (tabela pai):

```
CREATE TABLE clientes (
    id INT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL
);
```

2. Criar a tabela pedidos (tabela filha) com chave estrangeira:

```
CREATE TABLE pedidos (
    id INT PRIMARY KEY,
    cliente_id INT,
    data_pedido DATE,
    FOREIGN KEY (cliente_id) REFERENCES clientes(id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

Alguns pontos importantes:

- cliente_id em pedidos faz referência ao id da tabela clientes.

- ON DELETE CASCADE: se um cliente for deletado, todos os pedidos dele também serão deletados automaticamente.
- ON UPDATE CASCADE: se o id do cliente for alterado, o cliente_id nos pedidos será atualizado automaticamente.

2.6. Comando *SHOW TABLES*

Para visualizar todas as tabelas em um banco de dados, utilize o comando *SHOW TABLES.*

Exemplo:

```
SHOW TABLES;
```

2.7. Comando DESC

Para visualizar a estrutura de uma tabela, utilize o comando DESC (ou DESCRIBE).

Exemplo:

```
DESC clientes;
```

3. Manipulando dados das tabelas (DML – *Data Manipulation Language*)

Os comandos DML permitem realizar operações de inserção, alteração, exclusão e seleção sobre os registros (linhas) das tabelas. Comandos que compõem a DML: *INSERT, UPDATE, DELETE* e *SELECT*. Alguns autores definem que o comando *SELECT* faz parte de uma subdivisão chamada DQL (*Data Query Language* – Linguagem de Consulta de Dados).

Para exemplificarmos os comandos da DML, vamos utilizar a estrutura da tabela a seguir:

```
CREATE TABLE clientes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100),
    email VARCHAR(100),
    telefone VARCHAR(20),
    cidade VARCHAR(50)
);
```

3.1. Comando *INSERT*

O comando INSERT é usado para **adicionar novos registros** (linhas) em uma tabela do banco de dados.

Sintaxe:

INSERT INTO destino (coluna1, coluna2, ...)
VALUES (valor1, valor2, ...);

Onde;

- **Destino -** O nome da tabela ou consulta em que os registros devem ser anexados.
- **coluna1, coluna2** - Os nomes dos campos aos quais os dados devem ser anexados
- **valor1, valor2** - Os valores para inserir em campos específicos do novo registro. Cada valor é inserido no campo que corresponde à posição do valor na lista: Valor1 é inserido no campo1 do novo registro, valor2 no campo2 e assim por diante.

Os valores devem ser separados com uma vírgula e os campos de textos entre aspas duplas ou simples, você pode inserir:

- Uma linha por vez
- Várias linhas de uma vez
- Todos os campos ou apenas alguns

Exemplo:

```
INSERT INTO clientes (nome, email, telefone, cidade)
VALUES ('Ana Silva', 'ana.silva@email.com', '(11) 91234-5678', 'São Paulo');
```

Esse comando adiciona uma nova linha com os dados da Ana Silva. O campo id será preenchido automaticamente por ser AUTO_INCREMENT.

Inserindo múltiplos registros:

```
INSERT INTO clientes (nome, email, telefone, cidade)
VALUES
('João Lima', 'joao.lima@email.com', '(11) 99876-5432', 'Campinas'),
('Maria Souza', 'maria.souza@email.com', '(11) 98765-4321', 'Santos');
```

3.2 Comando *SELECT*

O comando SELECT é usado para **buscar dados** de uma ou mais tabelas no banco de dados. Ele permite filtrar, ordenar, agrupar e até realizar cálculos sobre os dados.

Sintaxe:
SELECT coluna1, coluna2, ... FROM nome_da_tabela;

Exemplo 1:
Buscar todos os clientes.

```
SELECT * FROM clientes;
```

Exemplo 2:
Buscar apenas nome e cidade.

```
SELECT nome, cidade FROM clientes;
```

Exemplo 3:
Filtrar clientes de São Paulo.

```
SELECT * FROM clientes
WHERE cidade = 'São Paulo';
```

3.3 Comando *UPDATE*

O comando UPDATE é usado para **modificar dados existentes** em uma ou mais colunas de uma ou mais linhas de uma tabela.

Sintaxe:
UPDATE nome_da_tabela
SET coluna1 = valor1, coluna2 = valor2, ...
WHERE condição

onde:
- **nome_da_tabela** - O nome da tabela cujos dados você quer modificar.
- **Valor1** - Uma expressão que determina o valor a ser inserido em um campo específico nos registros atualizados.
- **condição** - Uma expressão que determina quais registros devem ser atualizados. Só os registros que satisfazem a expressão são atualizado.

Exemplo:

```
UPDATE clientes
SET telefone = '(11) 90000-0000'
WHERE id = 1;

#ou

UPDATE clientes
SET nome = 'Ana Paula Silva', email = 'ana.paula@email.com'
WHERE id = 2;
```

Dicas importantes

- Sempre revise a cláusula WHERE antes de executar o comando.
- Você pode testar antes com um SELECT usando a mesma condição para ver quais registros serão afetados.
- Se você omitir o WHERE, **todos os registros da tabela serão atualizados**.