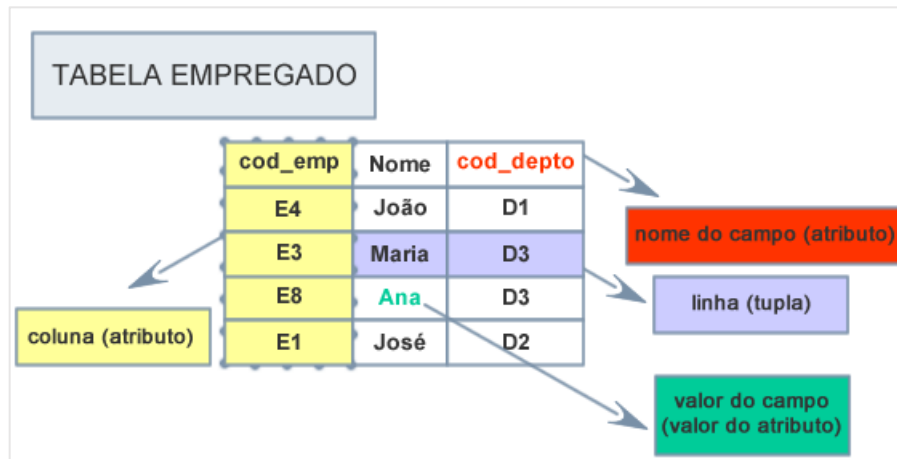


# SUMÁRIO

1.	ESTRUTURA DE UM BANCO DE DADOS.....	1
2.	CRIAÇÃO E EXCLUSÃO DE BANCOS DE DADOS .....	1
2.1.	CRIANDO UM BANCO .....	2
2.2.	EXCLUINDO UM BANCO .....	2
3.	CRIAÇÃO DE TABELAS, ATRIBUTOS E CONSTRAINS .....	3
4.	EDITANDO NOME DE TABELAS .....	3
5.	INCLUIR, EDITAR E REMOVER COLUNAS DE TABELAS .....	4
5.1.	ADICIONAR UMA COLUNA .....	4
5.2.	EDITAR UMA COLUNA .....	5
5.2.1.	CHANGE COLUMN.....	5
5.2.2.	MOFIDY COLUMN .....	5
5.3.	REMOVENDO COLUNAS .....	5
6.	COMANDO INSERT – INSERINDO REGISTROS .....	6
7.	COMANDO SELECT – CONSULTANDO DADOS .....	7
7.1.	SELETORES ADICIONAIS .....	7
8.	COMANDO UPDATE.....	8
9.	COMANDO DELETE .....	8
9.1.	COMANDO TRUNCATE.....	8
10.	RELACIONAMENTO DE TABELAS.....	9
10.1.	PRIMARY KEY E FOREGN KEY .....	9
10.2.	TIPOS DE RELACIONAMENTOS.....	10
10.2.1.	UM PARA MUITOS (1:N).....	11
10.2.2.	MUITOS PARA MUITOS (N:N) .....	11
10.3.	RELACIONAR TABELAS COM SQL .....	12
11.	PESQUISA AVANÇADA – JOINS .....	13
11.1.	LEFT JOIN .....	13
11.2.	RIGHT JOIN .....	14
11.3.	INNER JOIN.....	14
12.	FONTES .....	15

## 1. ESTRUTURA DE UM BANCO DE DADOS

Um banco de dados é uma forma de organização de conjuntos de arquivos relacionados entre si, dispostos em linhas, também chamadas de tuplas ou registros, e colunas, também chamadas de atributos ou campos.

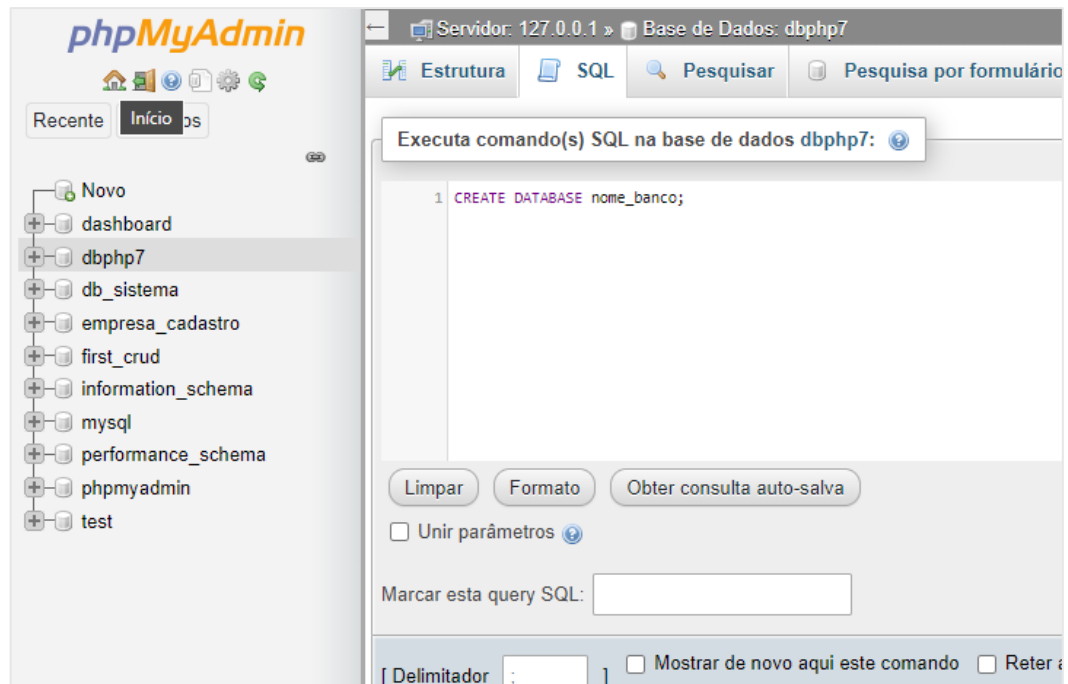


## 2. CRIAÇÃO E EXCLUSÃO DE BANCOS DE DADOS

A prática será explicada com aplicações dentro do ambiente do software phpMyAdmin, e com prioridade para a construção por meio da linguagem SQL. No entanto, para deixar claro, o uso da linguagem não é obrigatório, e a construção e coordenação do banco pode ser realizado por meio de ferramentas diretas do software.

Os exemplos serão apresentados por meio do uso da linguagem, na maioria dos casos, devido a sua importância e uso recorrente em qualquer ambiente profissional.

## 2.1. CRIANDO UM BANCO



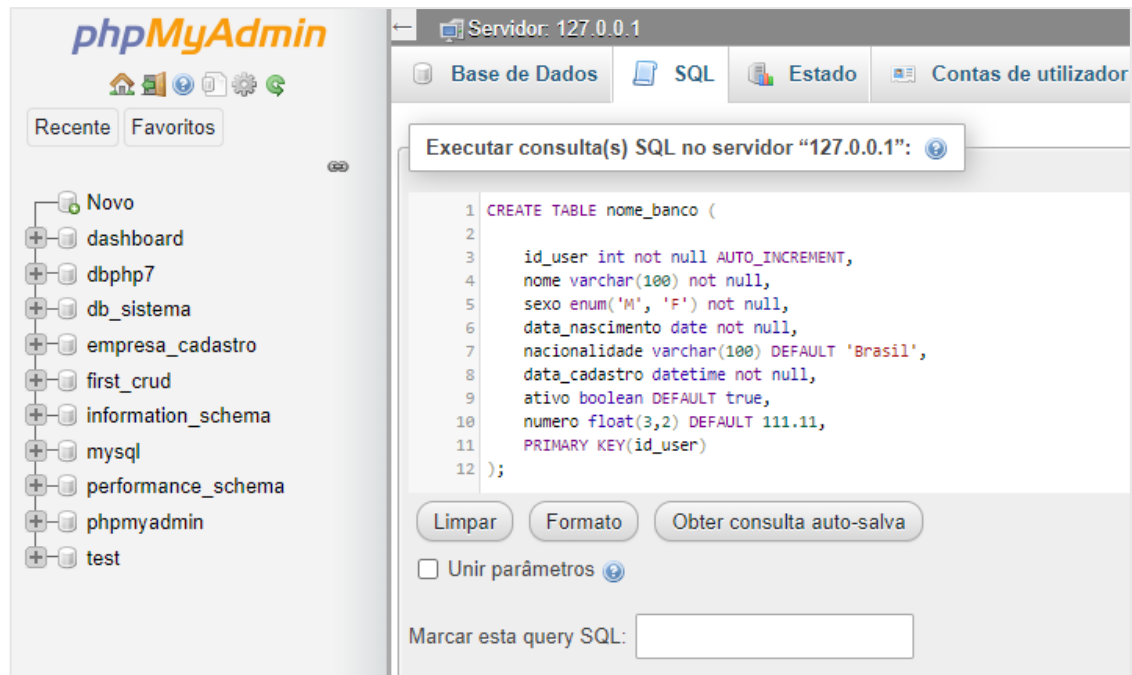
**Criar um banco é diferente da criação de uma tabela**, que são os conjuntos de dados dentro do banco. O primeiro passo, portanto, como realizado na imagem, é a criação do banco que irá conter tabelas diversas, posteriormente.

Na mesma imagem, a esquerda, podemos ver uma lista no ambiente phpMyAdmin, que é a lista de bancos de dados criados, e armazenados no servidor.

## 2.2. EXCLUINDO UM BANCO

**Para excluir um banco de dados**, inteiramente, utiliza-se o comando SQL “DROP DATABASE nome;”.

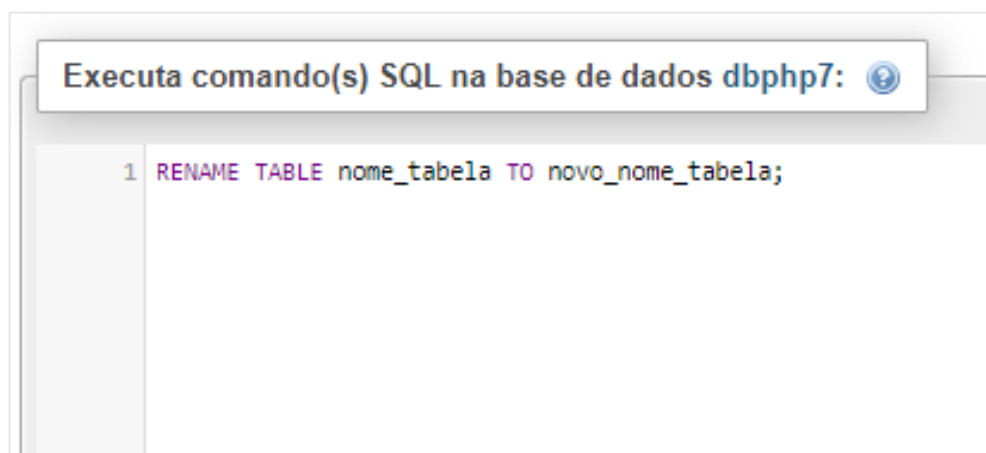
### 3. CRIAÇÃO DE TABELAS, ATRIBUTOS E CONSTRAINS



Os atributos da tabela, como nome, nacionalidade, observáveis na imagem acima, são os nomes das colunas, e as definições após são as constrains, isto é, são as configurações para o tipo de dado que determinada coluna terá.

Por exemplo, a coluna “id\_user” receberá um dado do tipo inteiro, não admite nulo, e a cada registro novo seu valor é auto incrementado. Já a coluna “nome” é do tipo varchar(100), ou seja, receberá strings com tamanho máximo de 100 caracteres, além de também ter sido configurada como “not null”, para não admitir o valor nulo.

### 4. EDITANDO NOME DE TABELAS



## 5. INCLUIR, EDITAR E REMOVER COLUNAS DE TABELAS

### ALTER TABLE

**ADD** - Permite a inclusão de uma nova coluna em um tabela.

**CHANGE** - Permite a alteração do nome de uma coluna e de suas propriedades, como por exemplo o tipo.

**DROP** - Permite a remoção de uma coluna da tabela.

### 5.1. ADICIONAR UMA COLUNA

A coluna é adicionada à tabela respectivamente selecionada, e também, no próprio comando, é possível modelar o tipo de dado que irá conter, isto é, definir as constrains da coluna.

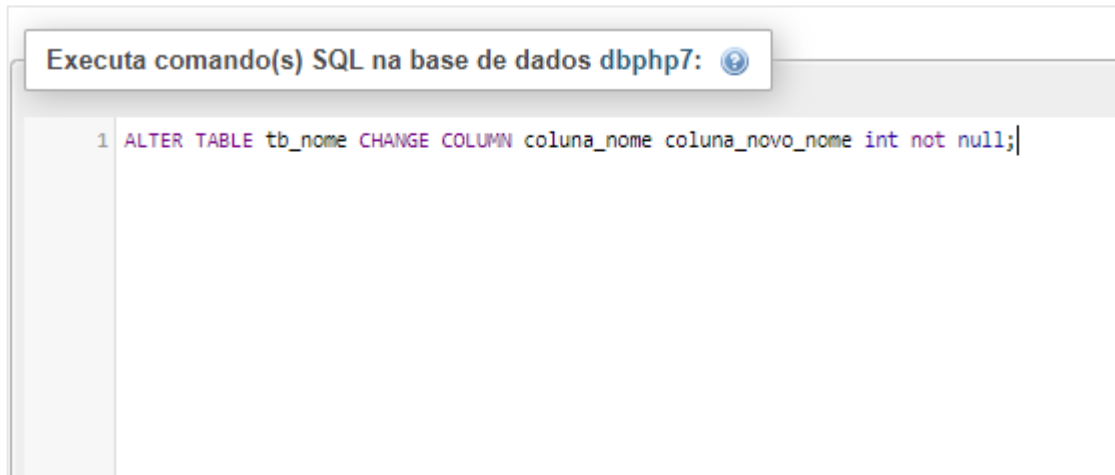
Executa comando(s) SQL na base de dados dbphp7: ?

```
1 ALTER TABLE tb_nome ADD COLUMN nome_coluna varchar(10) not null;
```

## 5.2. EDITAR UMA COLUNA

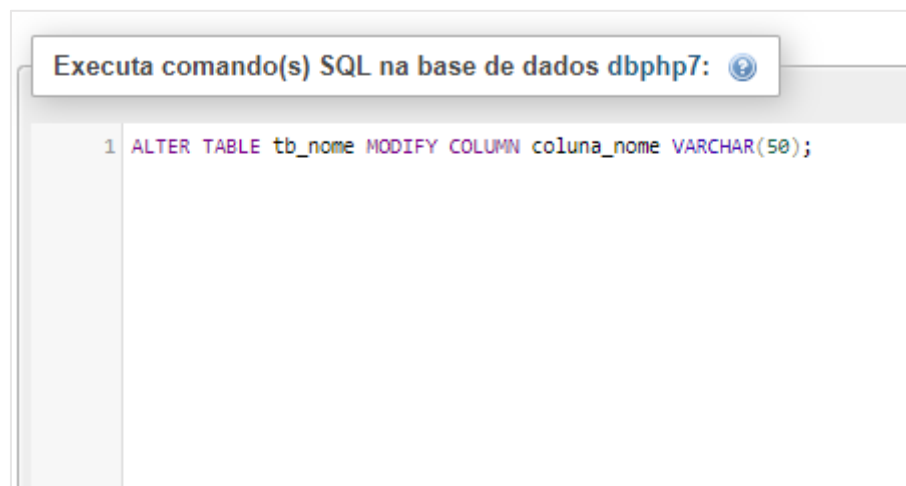
### 5.2.1. CHANGE COLUMN

O comando “**change column**” permite alterar o nome as definições da coluna, como demonstrado na imagem abaixo.

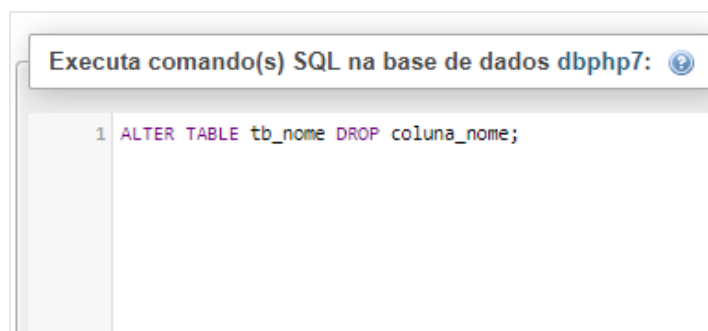


### 5.2.2. MOFIDY COLUMN

O comando “**modify column**” permite apenas alterar as definições da coluna, como demonstrado na imagem abaixo.

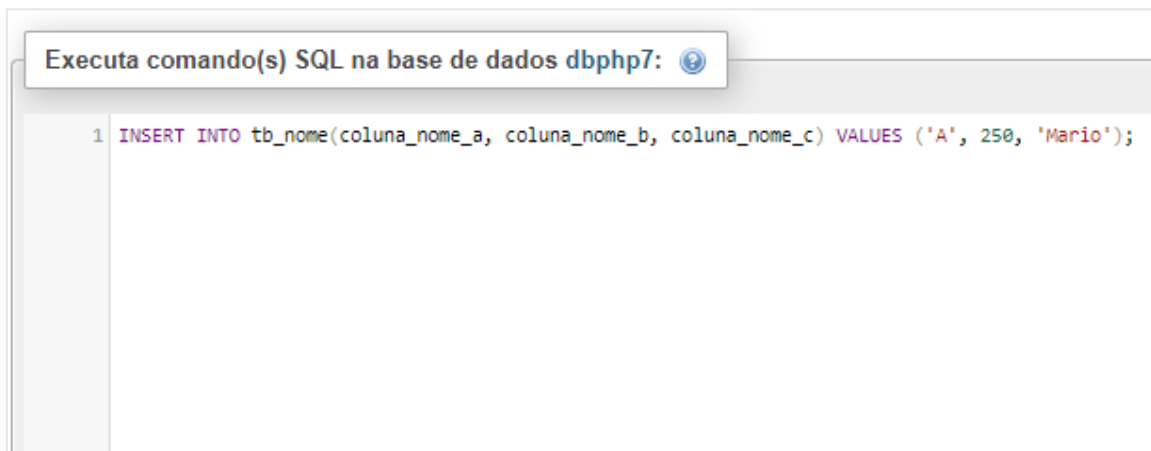


## 5.3. REMOVENDO COLUNAS



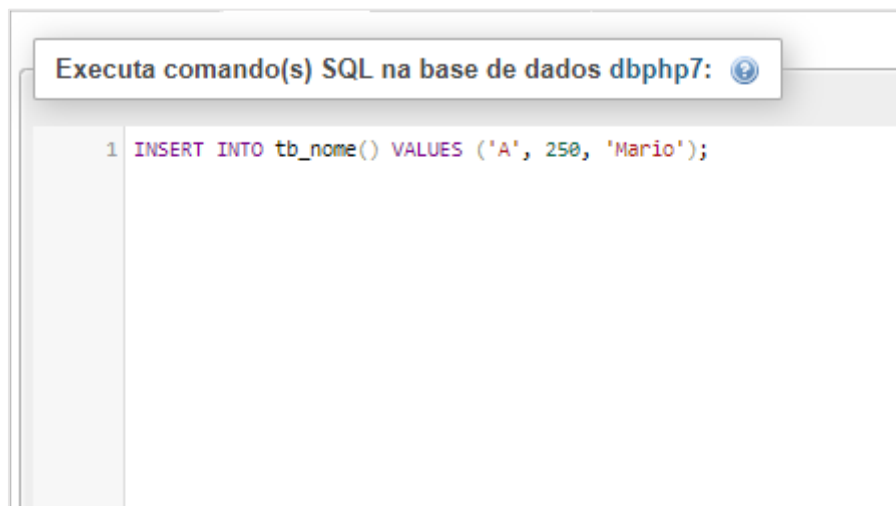
## 6. COMANDO INSERT – INSERINDO REGISTROS

Como demonstrado abaixo, os valores são inseridos nas colunas respectivamente, isto é, na ordem como descritos, assim, a primeira coluna recebe o primeiro dado 'A', a segunda coluna recebe o segundo dado, que é o número 250, e a terceira coluna recebe o terceiro dado 'Mario'.



A screenshot of a SQL command window titled "Executa comando(s) SQL na base de dados dbphp7:". The window contains a single line of SQL code: `1 INSERT INTO tb_nome(coluna_nome_a, coluna_nome_b, coluna_nome_c) VALUES ('A', 250, 'Mario');`. The code is color-coded: '1' is purple, 'INSERT' is green, 'INTO' is blue, 'tb\_nome' is red, the column names are black, 'VALUES' is green, and the values are in single quotes.

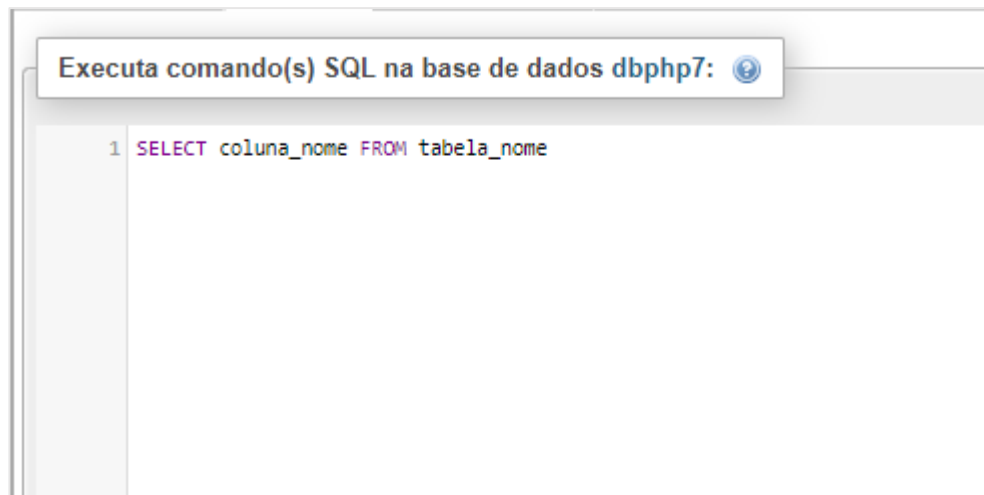
Outra forma de fazer exatamente a mesma coisa é sem definir o nome das colunas, pois assim, os valores descritos serão inseridos de acordo com a ordem das colunas em cada linha. Veja abaixo o mesmo exemplo de cima, mas realizado desta outra forma:



A screenshot of a SQL command window titled "Executa comando(s) SQL na base de dados dbphp7:". The window contains a single line of SQL code: `1 INSERT INTO tb_nome() VALUES ('A', 250, 'Mario');`. The code is color-coded: '1' is purple, 'INSERT' is green, 'INTO' is blue, 'tb\_nome' is red, the empty parentheses are black, 'VALUES' is green, and the values are in single quotes.

## 7. COMANDO SELECT – CONSULTANDO DADOS

O comando SELECT permite recuperar os dados de um objeto do banco de dados, como uma tabela, view e, em alguns casos, uma stored procedure (alguns bancos de dados permitem a criação de procedimentos que retornam valor). A sintaxe mais básica do comando é:



O caractere \* representa todos os campos:



### 7.1. SELETORES ADICIONAIS

O comando básico para procura de dados é o expresso nas imagens acima, em que claramente as buscas são genéricas, podendo e de fato retornando diversos dados se existirem. É possível, no entanto, realizar buscar por dados específicos, a partir da definição de mais critérios para a busca.

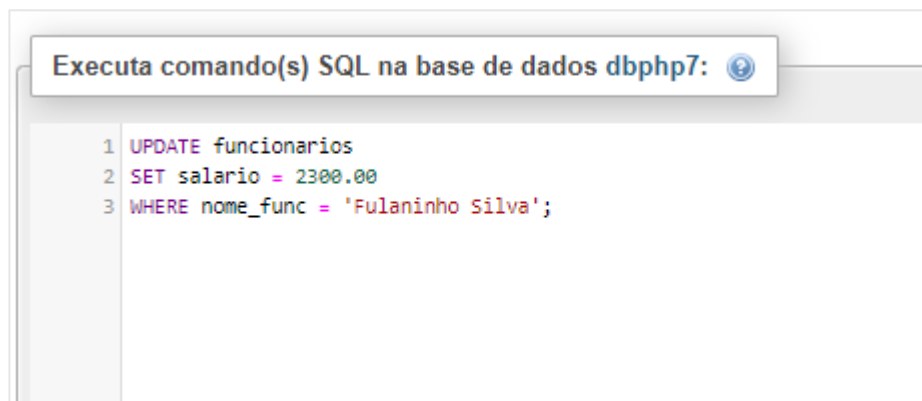
Ver em: <https://www.devmedia.com.br/sql-select-guia-para-iniciantes/29530>



## 8. COMANDO UPDATE

O comando para atualizar registros é UPDATE. Esse comando permite atualizar os dados em múltiplos registros de uma vez, mas somente opera em uma tabela por vez.

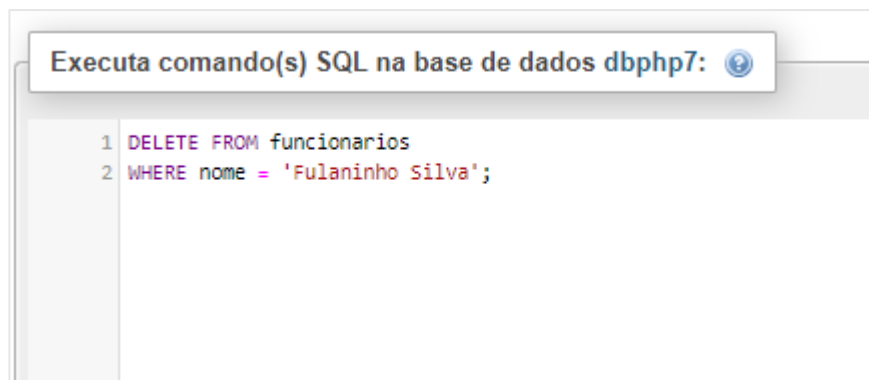
A **palavra-chave SET** é usada para definir qual **coluna** será modificada, assim como o novo valor a ser atribuído a ela. A **cláusula WHERE**, apesar de opcional, é de suma importância para o comando UPDATE. Se ela não for usada, toda a tabela será atualizada – todos os registros. Já com a cláusula WHERE, somente os registros que correspondam ao filtro aplicados serão atualizados.



```
1 UPDATE funcionarios
2 SET salario = 2300.00
3 WHERE nome_func = 'Fulaninho Silva';
```

## 9. COMANDO DELETE

O comando utilizado para apagar dados é o DELETE, e deve sempre ser utilizado acompanhado da cláusula WHERE, pois, do contrário, todos os registros da tabela serão excluídos. Veja um exemplo simples abaixo:



```
1 DELETE FROM funcionarios
2 WHERE nome = 'Fulaninho Silva';
```

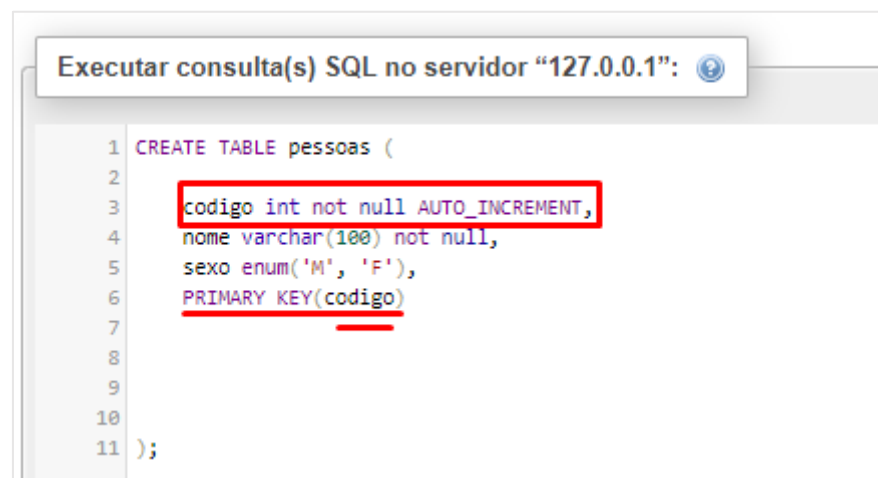
### 9.1. COMANDO TRUNCATE

Este comando é igual ao comando DELETE, mas sem a especificação WHERE, ou seja, permite a exclusão de todos os dados de uma tabela a partir de uma única execução.

## 10. RELACIONAMENTO DE TABELAS

### 10.1. PRIMARY KEY E FOREIGN KEY

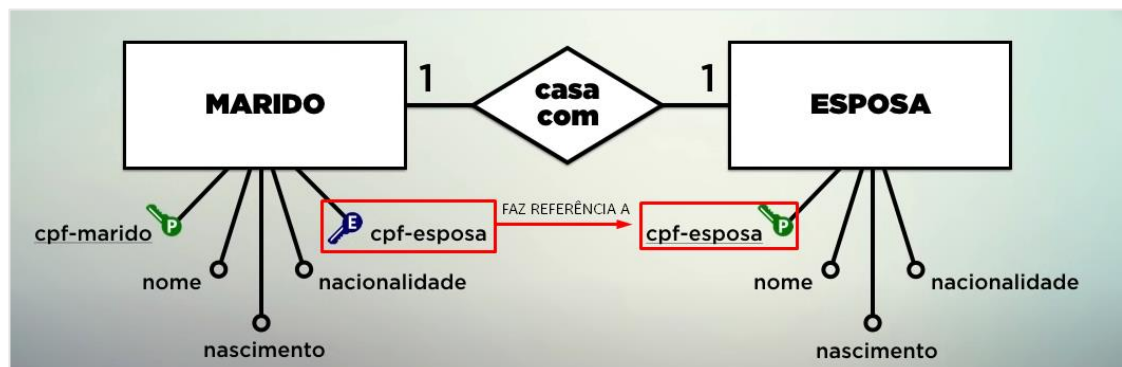
A **chave primária** é um tipo de dado que irá servir para dar identidade a uma linha, sendo, portanto, única, e, por via de regra, não nula. É utilizada, dada sua utilidade, para, por exemplo, a busca de dados, isto é, serve como um referencial principal para a seleção de registros. É comum que seja usado um identificador numérico sequencial para isso, ou seja, um dado INT, com as constrains NOT NULL e AUTO INCREMENT. Veja abaixo um exemplo:



```

Executar consulta(s) SQL no servidor "127.0.0.1":
1 CREATE TABLE pessoas (
2
3     codigo int not null AUTO_INCREMENT,
4     nome varchar(100) not null,
5     sexo enum('M', 'F'),
6     PRIMARY KEY(codigo)
7
8
9
10
11 );
  
```

Por outro lado, a **chave estrangeira** é, na maioria dos casos, uma chave primária vinda de outra tabela, servindo, portanto, para o estabelecimento de relações entre diferentes tabelas. Por exemplo, se existe uma tabela de alunos, e uma de cursos, poderíamos adicionar uma coluna chamada “curso preferido” na tabela dos alunos, e defini-la como uma FOREIGN KEY que faz referência ao atributo “código” da tabela de cursos. Note que a chave estrangeira não precisa ter o exato mesmo nome daquela a que faz referência, mas sim a mesma definição para o tipo de dado armazenamento, é claro. Veja abaixo um exemplo:



Na imagem acima, a relação se dá entre a entidade “Marido”, e a entidade “Esposa”. Para estabelecer uma relação deste tipo, de um para um, basta que apenas uma das entidades faça referência a outra, pois cada uma delas só pode ser “apontada” por uma única outra. Assim, no caso daquele exemplo, a entidade “Marido” é que está fazendo a referência, ou estabelecendo a relação, e por conseguinte, a chave estrangeira pertencerá a ele.

A entidade “Esposa” possui uma chave primária, que é o seu CPF, e assim, para a chave estrangeira da entidade “Marido”, será passado o valor do CPF da esposa por referência. Isto é, como uma típica passagem de valor por referência, a entidade “Marido”, por meio de sua chave estrangeira, “aponta” para o CPF de sua esposa, cujo valor é único.

Assim percebe-se que para criar relacionamentos entre tabelas os dados que conversam não são enviados em si para estabelecer a relação, ao invés disso, são utilizados atributos cuja única função, neste contexto, é servir de referencial para entidades externas, como uma espécie de “ponteiro”.

## 10.2. TIPOS DE RELACIONAMENTOS

No exemplo anterior, citei que o relacionamento era do tipo **“um para um”**, mas, naturalmente, no mundo real existem diversas formas de relacionamentos, em que as entidades não necessariamente devem se relacionar com uma outra exclusivamente.

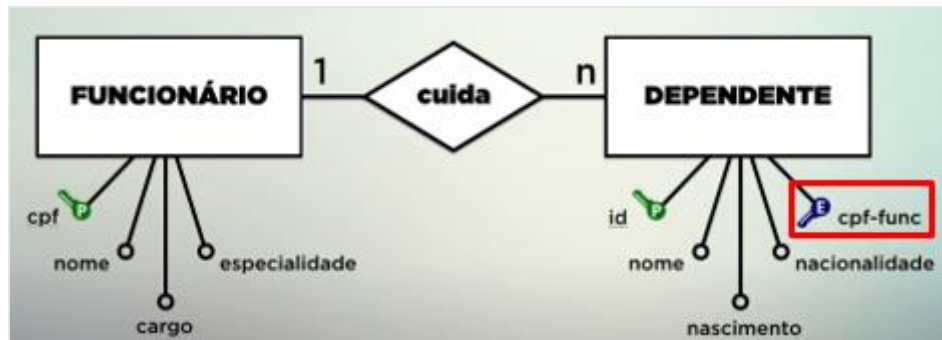
**Poderíamos citar o tipo “um para muitos”**. A exemplo, um mesmo cliente chamado “Fulano”, em uma loja, pode comprar um, ou mais produtos, e assim sua relação, ao invés de um para um, será de um – cliente- para muitos- produtos-.

**Poderíamos citar também o tipo “muitos para muitos”**. A exemplo, novamente em uma loja, a partir de uma perspectiva geral, e sem selecionarmos um único cliente, podemos afirmar que os produtos podem ser comprados por diversos clientes, ou podemos ter diversos clientes comprando diversos produtos. Mas como estabelecer uma relação entre registros, e identificar qual cliente comprou os produtos, com esta perspectiva geral? Devido a este problema, surge um diferencial na forma como é realizado este relacionamento, que será demonstrada em breve.

**As regras para passagem de chave estrangeira**, apesar de serem lógicas, seguem alguns padrões que dispensam, em diversos casos, um entendimento para a decisão de onde aloca-la. Veja nos próximos tópicos os tipos de relacionamento e forma de alocação das chaves.

### 10.2.1. UM PARA MUITOS (1:N)

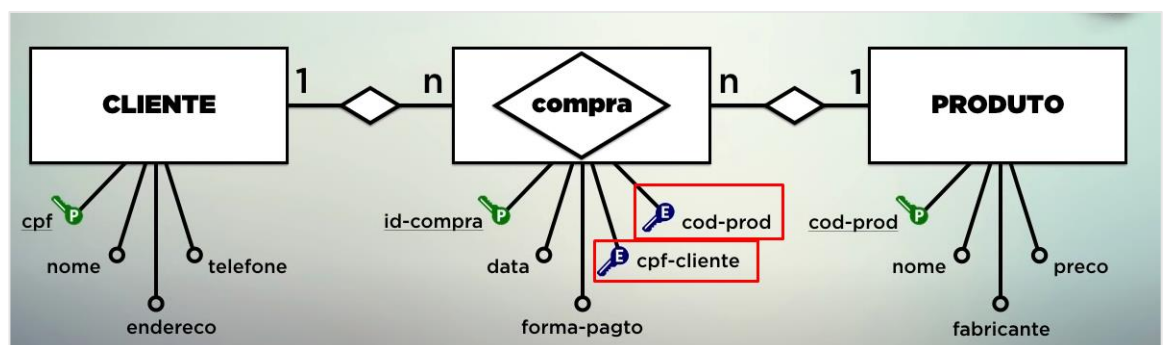
Como regra geral, neste tipo de relacionamento a chave estrangeira é criada dentro da entidade “muitos”. Veja o exemplo:



### 10.2.2. MUITOS PARA MUITOS (N:N)

Como supracitado, no tópico 10.2, este relacionamento possui um diferencial. Note que se a relação estabelecida na tabela fosse dada a partir de uma perspectiva geral, como de clientes para produtos, de muitos para muitos, seria impossível estabelecer as relações e identificar qual cliente comprou tal produto, ou qual registro da tabela de clientes se relacionou com o registro do produto, na tabela de produtos. Na prática do mundo real, quando produtos são comprados por alguém, a relação é de um para muitos.

Por isso, neste caso, o próprio diagrama de relacionamento sofre uma mudança. Ao invés de, entre as duas entidades que se relacionam, haver um nó de relacionamento, haverá uma terceira entidade, que servirá como intermédio para o realizar das especificação das relações.



Assim, muitos clientes podem comprar muitos produtos, a partir de uma vista geral, mas, as relações, na prática, serão sempre de um para muitos, em que cada cliente vale por um, e cada produto comprado, igualmente, vale por um.

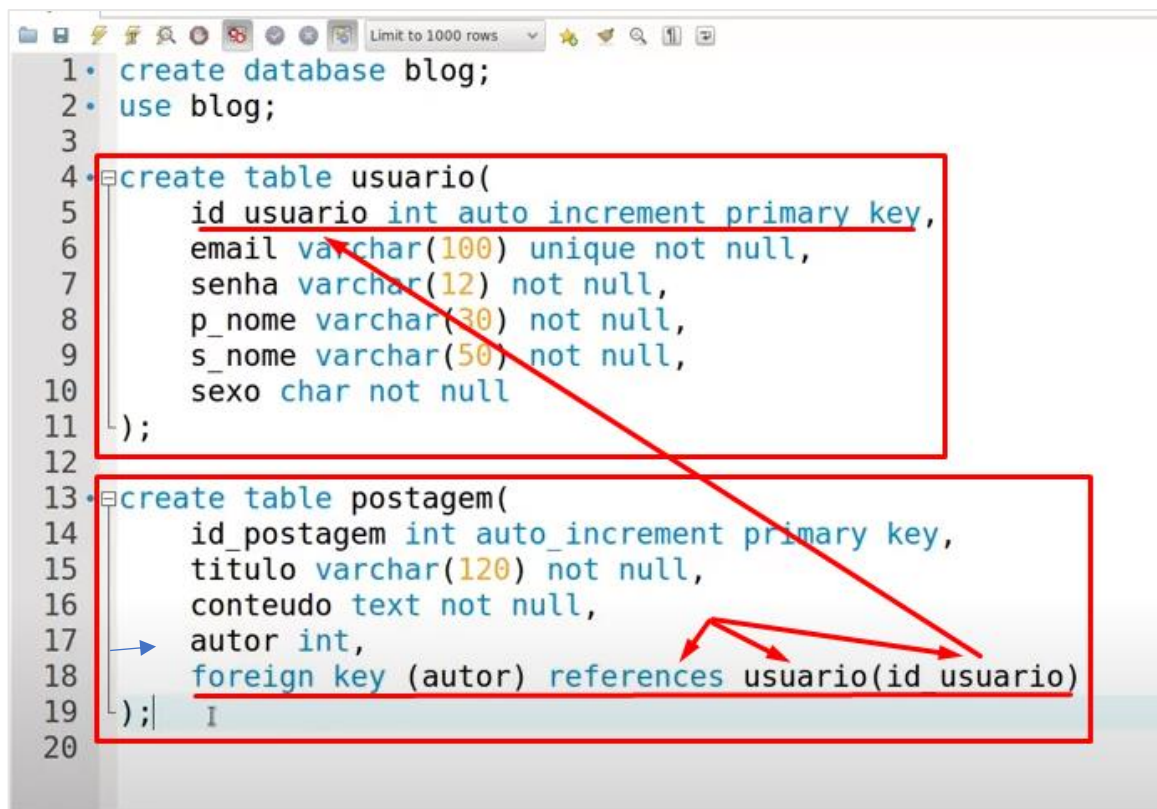
**Para a passagem da chave estrangeira, vale a regra do um para muitos:** a chave é definida no lado “muitos”. Por isso, neste caso, a entidade intermediária da relação terá duas chaves estrangeiras, cada uma fazendo referência à chave primária de cada uma das outras duas entidades.

### 10.3. RELACIONAR TABELAS COM SQL

Em linguagem SQL, a sintaxe de definição da chave estrangeira segue o mesmo padrão inicial da chave primária. Veja que o nome do atributo que servirá de chave é escrito entre parênteses.

Em seguida, para estabelecer a referência, o comando de definição da chave estrangeira é seguida da clausula “references”, que implica em “faz referência a”, e após este termo é escrito o nome tabela com o atributo referenciado, desta mesma tabela, entre parênteses.

Em outras palavras, o comando significa: “A chave estrangeira será (Y atributo desta tabela), que fará referencia a outra tabela, e mais especificamente (ao valor do seu atributo X)”.



```

1 • create database blog;
2 • use blog;
3
4 • create table usuario(
5     id_usuario int auto increment primary key,
6     email varchar(100) unique not null,
7     senha varchar(12) not null,
8     p_nome varchar(30) not null,
9     s_nome varchar(50) not null,
10    sexo char not null
11 );
12
13 • create table postagem(
14     id_postagem int auto_increment primary key,
15     titulo varchar(120) not null,
16     conteudo text not null,
17     autor int,
18     foreign key (autor) references usuario(id_usuario)
19 );
20

```

A chave estrangeira pode ser adicionada após a criação da tabela, por meio do comando ALTER TABLE. Veja:

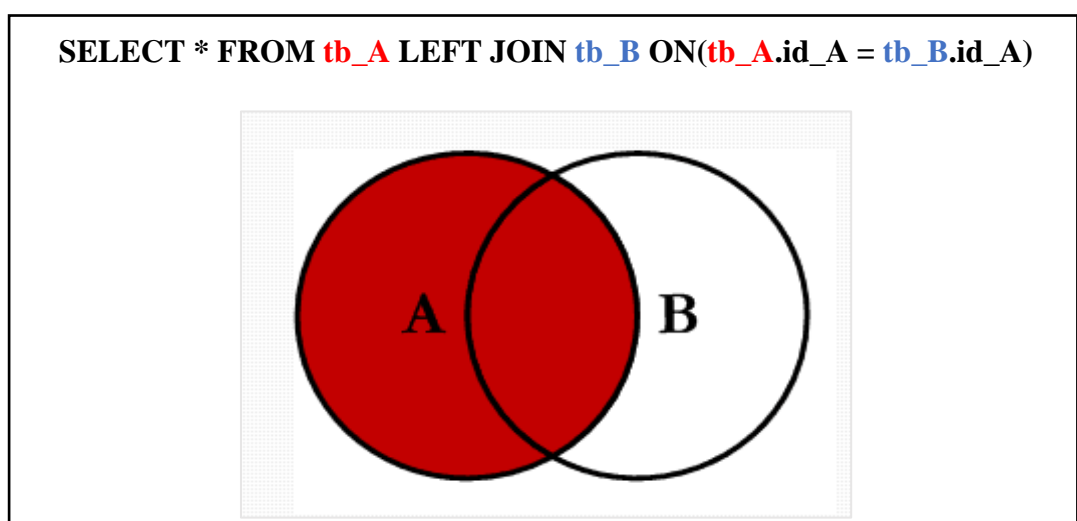
```
Executar consulta(s) SQL no servidor "127.0.0.1":  
1 ALTER TABLE pessoas add curso_preferido int;  
2  
3 alter table pessoas add FOREIGN KEY (curso_preferido) REFERENCES cursos(id_curso);  
4
```

## 11. PESQUISA AVANÇADA – JOINS

A abordagem desta clausula é a de instruções SELECT avançadas, que tragam o resultado de duas ou mais tabelas, isto é, façam joins entre estas tabelas e retorne seus dados num único comando. Portanto, junções, obtidas por JOINS, permitem buscas e retornos de dados de diferentes tabelas.

### 11.1. LEFT JOIN

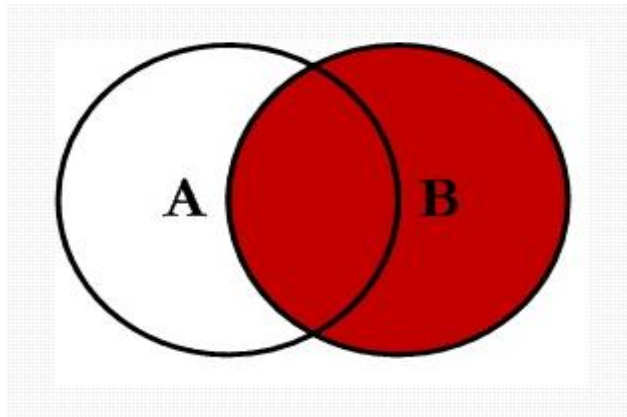
O **Left Join**, cujo funcionamento é ilustrado na Figura ABAIXO, tem como resultado todos os registros que estão na tabela A (mesmo que não estejam na tabela B) e os registros da tabela B que são comuns à tabela A.



Em outras palavras, no comando SQL, a tabela da esquerda será prioritária em relação a tabela da direita, e da primeira serão mostrados todos os registros, enquanto da segunda apenas aqueles que atenderem a condição de relacionamento estabelecida em ON().

### 11.2. RIGHT JOIN

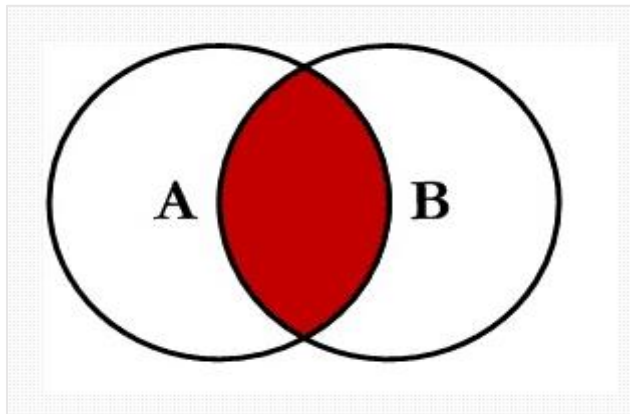
Usando o **Right Join**, conforme mostra a Figura abaixo, teremos como resultado todos os registros que estão na tabela B (mesmo que não estejam na tabela A) e os registros da tabela A que são comuns à tabela B. Ou seja, este caso é o mesmo do anterior, mas com a prioridade sendo em relação à tabela da direita, e não da esquerda.



### 11.3. INNER JOIN

O **Inner Join** é o método de junção mais conhecido e, como ilustra a Figura 2 abaixo, retorna os registros que são comuns às duas tabelas.

```
SELECT * FROM tb_A INNER JOIN tb_B ON(tb_A.id_A = tb_B.id_A)
```



Neste caso, em contraste com os anteriores, serão retornados, de ambas as tabelas, apenas os registros que estiverem de acordo com a relação definida em ON().

## 12. FONTES

<https://www.devmedia.com.br/comandos-basicos-em-sql-insert-update-delete-e-select/37170#:~:text=O%20comando%20utilizado%20para%20apagar%20dados%20%C3%A9%20o%20DELETE.>

<https://www.devmedia.com.br/comandos-basicos-em-sql-insert-update-delete-e-select/37170>

<http://www.bosontreinamentos.com.br/curso-completo-de-mysql/>

<https://www.youtube.com/watch?v=MNHDpqTFbFE>

[https://www.youtube.com/playlist?list=PLHz\\_AreHm4dkBs-795Dsgvau\\_ekxg8g1r](https://www.youtube.com/playlist?list=PLHz_AreHm4dkBs-795Dsgvau_ekxg8g1r)

<https://www.udemy.com/course/web-completo/>

<https://www.devmedia.com.br/sql-join-entenda-como-funciona-o-retorno-dos-dados/31006>