



## Sumário

Oque é uma chave primaria? .....	2
Oque é chave estrangeira (Foreing Key)? .....	2
Conhecendo um pouco sobre a ferramenta MySQL Workbench.....	2
Iniciando nos comandos SQL.....	3
Acessando o MySQL por linha de comando .....	6
Criando tabelas.....	7
Inserir itens na tabela .....	11
Alterar dados de uma tabela .....	11
Exclusão de itens dentro do SQL .....	12
Variações do comando Alter Table .....	12
Utilizando o SELECT para pesquisas.....	14
Agrupando e Agregando .....	17



# Introdução ao SQL

## O que é uma chave primaria?

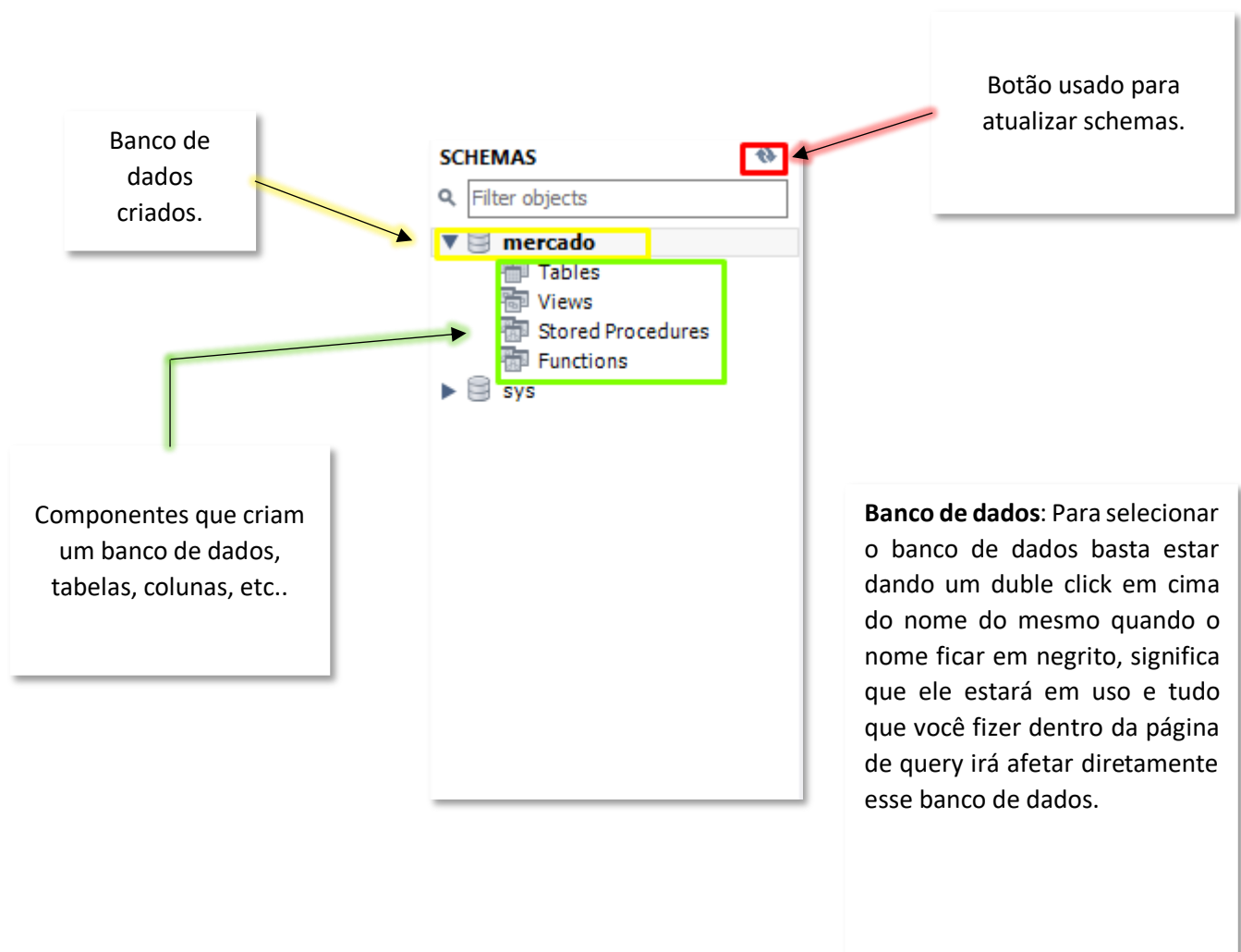
O valor armazenado em um atributo ou mais atributos de um registro deve ser único em relação a todos os registros da tabela.

## O que é chave estrangeira (Foreign Key)?

É a chave que permite a referência a registros oriundos de outras tabelas. Ou seja, é o campo ou conjunto de campos que compõem a chave primária de uma outra tabela. Uma chave estrangeira é a representação de um relacionamento entre tabelas.

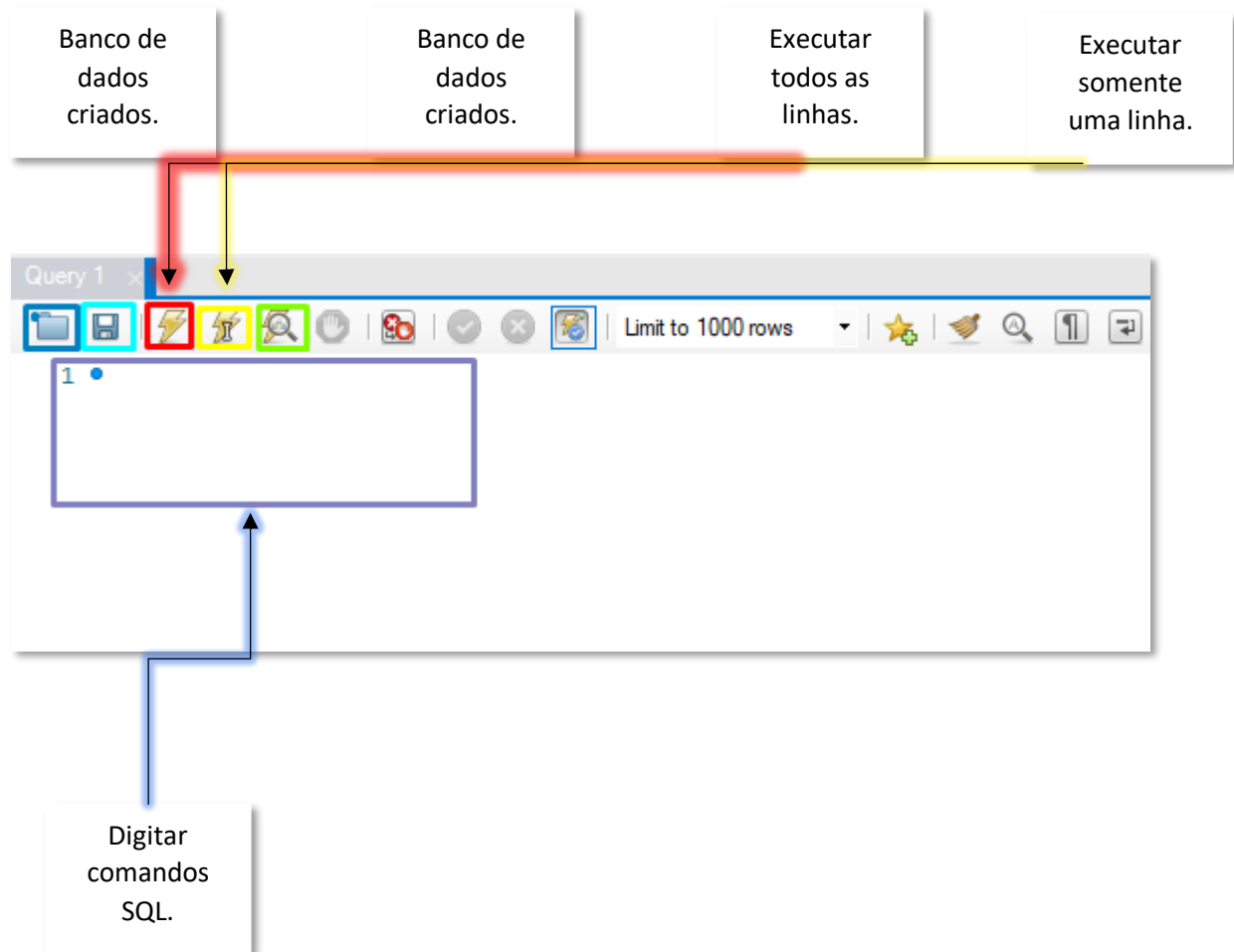
## Conhecendo um pouco sobre a ferramenta MySQL Workbench.

### 1. schemas





## 2. Query



- Ao digitar um comando nunca se esquecer do ponto de virgula no final, pois caso não tenho isso pode gerar um erro em seu código.
- Verificar se os nomes das tabelas estão corretos antes de executar um comando é sempre uma boa prática.

## Iniciando nos comandos SQL

### 1. Criando um banco de dados

- `CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name`

Podemos utilizar esses comandos para criar um banco de dados, sendo que database e schema são sinônimos.

No comando IF NOT EXISTS estamos passando para o MySQL que ele só irá criar o banco de dados se o mesmo ainda não existir, caso ele exista a IED não



realizara o comando.

➤ **Creat\_specification:**

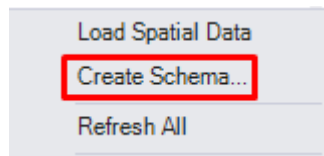
```
create_specification:  
    [DEFAULT] CHARACTER SET [=] charset_name  
    | [DEFAULT] COLLATE [=] collation_name  
    | DEFAULT ENCRYPTION [=] {'Y' | 'N'}
```

Temos que ajustar o banco de dados para que ele possa compreender nossa linguagem, assim como no HTML temos que especificar o UTF-8 no SQL não é diferente para isso inserimos os comandos a cima juntamente com o CREAT DATABASE.

➤ Podemos criar um Schema sem utilizar a query (Isso usando a IDE Workbench)

**Como realizar:**

Clique com o botão direito do mouse em uma área branca dentro de schemas e vá em Creat Schema.



Ele abrira uma tela para que você possa inserir as informações do banco de dados como nome, charset e collation.

A screenshot of a dialog box for creating a new schema. It has a title bar with a database icon. The 'Name' field contains 'new\_schema'. There is a 'Rename References' button. The 'Charset/Collation' section has two dropdown menus, both set to 'Default'.

Name:

Charset/Collation:



- Após isso clicar em Apply

Online DDL

Algorithm:  Lock Type:

1 CREATE SCHEMA `new\_schema` ;

2

<  >

Back Apply Cancel

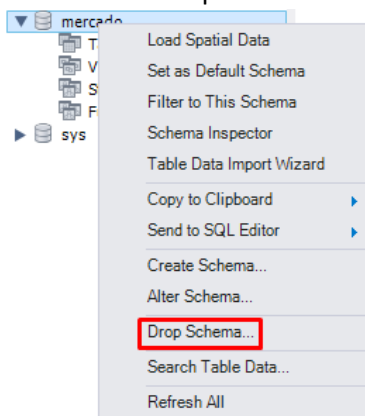
A IDE irá te mostrar o comando que será utilizado para criação deste banco, onde você pode além de aprender o comando agilizar o trabalho de criação.

- Podemos também apagar um banco de dados usando o seguinte comando:

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

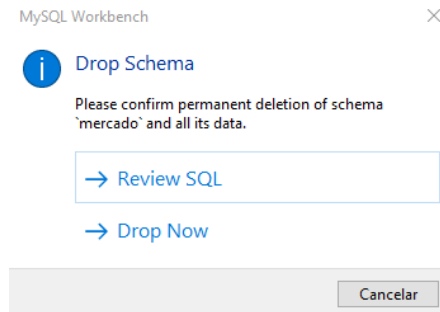
Uma outra forma de apagar um banco de dados é utilizando a IDE:

- Clicando com o botão direito do mouse em cima do nome do banco, localizado no campo de schemas.
- Clicando em Drop Schema.





- Aparecera duas informações em nossa tela onde a primeira (Review SQL) lhe mostrara o código antes da execução e a segunda (Drop Now) apagara o banco de dados.



## Acessando o MySQL por linha de comando

2. Para acessar o MySQL por linha de comando primeiro temos que navegar até a pasta onde o MySQL está instalado.

```
Pasta de C:\Program Files\MySQL\MySQL Workbench 8.0 CE
```

Este é o caminho este caso em específico.

3. Após isso precisando digitar o seguinte comando.

```
C:\Program Files\MySQL\MySQL Workbench 8.0 CE>mysql -h localhost -u root -p
```

Onde -h seria o host

O -u seria o usuário

E -p a senha, porém não vamos digitar a senha ainda somente após a execução dessa linha de comando.

4. Após a execução dessa tela será pedido a senha do banco de dados.

```
Enter password:
```

Agora podemos colocar a senha do nosso banco.

5. Após isso se tudo der certo estaremos dentro do ambiente do MySQL.

```
mysql>
```

6. Agora precisamos selecionar um banco de dados usando o comando:

```
mysql> use mercado;  
Database changed
```

7. Para sair do banco de dados utilizamos o comando EXIT.

```
mysql> exit  
Bye
```



## Criando tabelas

### 1. Tipos de campos:

#### ➤ Numéricos:

Tipo	Valor em Bytes	Menor Valor (Com Sinal) - Signed	Menor Valor (Sem sinal) - Unsigned	Maior Valor (Com sinal) - Signed	Maor valor (Sem sinal) - Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2xE63	0	2xE63-1	2xE64 - 1

- **Unsigned:** Significa que os números armazenados não terão sinais, dessa forma o campo comporta uma quantidade maior de caracteres a serem adicionados.

#### ➤ Ponto Flutuante:

- **Float** – Precisão simples (4 Bytes)  
Significa que se o SQL irá arredondar o número com casas decimais para o mais próximo possível.
- **Double** – precisão dupla (8 Bytes)  
Significa que a precisão desse campo será maior podendo também armazenar mais caracteres, se queremos um número o mais exato possível utilizamos o Double.
- Podemos também definir o quando de casas queremos que o número ocupe como por exemplo float(7,4), isso significa que esse número poderá ter 7 casas antes da vírgula e 4 casas depois da vírgula, porém podemos colocar número maiores o que acontecerá e que o SQL vai arredondar o número para caber dentro da formatação passada no comando.

#### ➤ Atributos dos campos numéricos:

- **SIGNED ou UNSIGNED** – vai possuir ou não sinal no número.
- **ZEROFILL – Preenche com Zeros os Espaços:**  
Exemplo: INT(4). Se armazenarmos o valor 5 será gravado 0005.



- **AUTO\_INCREMENT** – Sequencia auto incrementada.  
Exemplo: 1,2,3,4,5.....
- Erros de **OUT OF RANGE**:  
Vão ocorrer quando os valores estourarem os limites.

## 2. Data e Hora:

- **Date – 1000-01-01 até 9999-12-31.**  
O atributo Date é responsável por armazenar a data em um campo, lembro sempre dessa ordem ano > mês > dia.
- **DateTime – 1000-01-01 00:00:01 UTC até 2038-01-19 UTC.**  
Este atributo armazena data e hora dentro de um campo.
- **TimeStamp – 1970-01-01 00:00:01 UTC até 2038-01-19 UTC.**
- **Time - -838:59:59 e 839:59:59**
- **Year – 1901 – 2155 (Pode ser expresso no formato 2 ou 4 dígitos).**  
Armazenar o ano.

## 3. Strings:

- **Char – Cadeia de caracteres com valor fixo (de 0 a 255).**  
Isso significa que se definido que com campo vai ter dois caracteres mesmo estando vazio ele ocupará o espaço de dois caracteres.
- **Varchar – cadeia de caracteres com valor variado (de 0 a 255).**  
Significa que o campo só vai ocupar o tanto de caracteres que for inserido dentro do mesmo.
- **Binary – cadeia de caracteres com valor fixo (de 0 a 255).**  
Expressos em binário.
- **Varbinary – cadeia de caracteres com valor variado (de 0 a 255).**  
Expressos em binário.
- **Blob – Binário longo. Podemos ter:**
  - Tinyblob
  - Blob
  - Mediumblob
  - Longblob





➤ **Text – texto longo**

- Tinytext
- Text
- Mediumtext
- Longtext

➤ **Enum**

- Permite armazenar uma lista pré-definida de valores.
- Exemplo:
  - Size Enum("45", "42", "34", "40")Dessa forma o valor do campo Size pode ser somente alguma dessas opções.

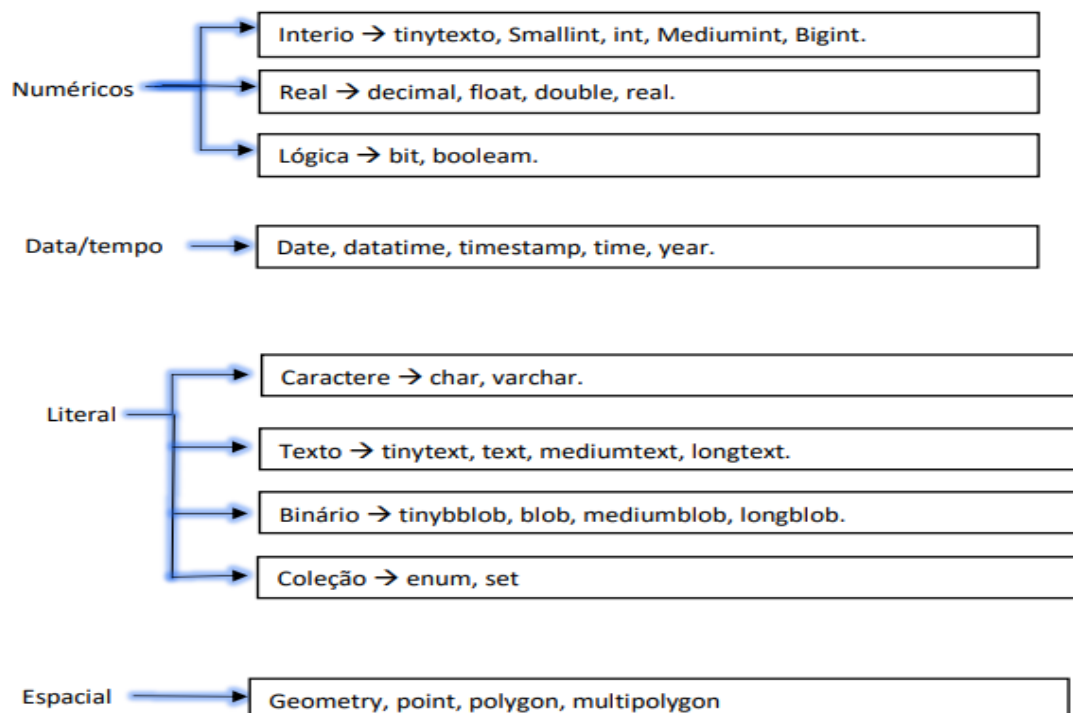
➤ **Atributos dos campos String:**

- Set e Collate – Que tipo de conjunto de caracteres serão suportados.

➤ **Spacial:**

- Geometry
- Point
- Linestring
- Polygon

### Tipos primitivos





➤ **Comando para criar uma tabela:**

- Para criar uma tabela no SQL usamos o seguinte comando:

```
5 • Create table tbCliente
6 (CPF varchar(11),
7 NOME varchar (100),
8 Endereco1 varchar(150),
9 Endereco2 varchar(150),
10 Bairro varchar(50),
11 Cidade Varchar(50),
12 Estado varchar(50),
13 CEP varchar(8),
14 Idade smallint,
15 Sexo enum("M", "F"),
16 Limite_Credito float,
17 Volume_Compra float,
18 Primeira_Compra bit(1)
19 );
```

- Create table Nome\_da\_tabela.
- Após isso abrimos parênteses e iniciamos com os nomes das colunas em seguida o tipo do campo.
- Ao termino não podemos esquecer de colocar a vírgula para iniciarmos a segunda coluna, é importante ressaltar que a última coluna não é necessária o termino com vírgula.
- Ao final do comando necessário fechar parênteses e colocar o ponto e vírgula.



## Inserir itens na tabela

- Para inserir dados em uma tabela usamos o comando Insert into como representado na imagem.

```
30 • insert into tbproduto (  
31     Produto, Nome, Embalagem, Tamanho, Sabor, Preço_Lista) value (  
32     '1040107', 'Light - 350 ml - Melância', 'Lata', '350 ml', 'Melância', 4.555  
33 );
```

- Insert into nome\_da\_tabela
- Logo em seguida devemos colocar os nomes das colunas (A ordem não importa).
- O parâmetro **value**.
- Tudo que for **string** deve estar entre aspas simples e tudo que for numérico não precisa dessa recomendação.

## Alterar dados de uma tabela

- Para alterar dados de uma tabela usamos o comando **UPDATE**

```
7 • update amigos set nome = 'guigui'  
8 where id = 1;
```

- Update nome\_da\_tabela.
  - Colocamos o set que se traduzido fica definir, logo após colocamos o nome da coluna juntamente com a informação nova.
  - Where que se traduzido fica onde. Basicamente estamos informando onde irá ocorrer a mudança dentro da tabela.
  - Colocando tudo de uma forma literal esse comando quer dizer: Altere a tabela amigos modificando a coluna nome para guigui onde o Id for igual a 1.
- Podemos também acrescentar o comando LIMIT, usando para limitar essa edição a somente um tanto de linhas específicas.



## Exclusão de itens dentro do SQL

- Temos três tipos de comando para exclusão de dados dentro de nosso SQL o primeiro que seria o **DROP**, o segundo **DELETE** e o terceiro **TRUNCATE**.

- Usamos o comando **DROP** quando queremos apagar itens dentro de nosso SQL.

**EX:** Bancos, tabelas e índices.

- `drop database cadastro;`
- `drop table cursos;`

- Usamos o comando **DELETE** quando queremos apagar componentes.

**EX:** linhas de uma tabela.

```
8  
9 • delete from cursos where nome = 'Curso de HTML5';  
10
```

Colocamos o comando **DELETE FROM** a tabela e nesse caso especificamos que linhas eu quero que ele apague.

- Usamos o comando **TRUNCATE** quando queremos remover todas as linhas de uma tabela sem registrar as exclusões de linhas individuais – ou seja, remove o conteúdo de uma tabela em uma única instrução compactador

## Variações do comando Alter Table

- O comando **ALTER TABLE** é usado para alterar propriedades da tabela como tipo do campo se ele é um tipo **INT** e passara a ser **VAR**, o nome do campo .....
- Usamos o **ALTER TABLE** em conjunto com alguns outros comandos para filtragem e pesquisa.
- A baixo vamos listar alguns:
  - **ADD COLUMN** responsável por adicionar uma nova coluna dentro de uma tabela.



```
5  alter table cursos
6  add column idcurso
```

- **DROP COLUMN** responsável por apagar uma coluna dentro e uma tabela.

```
3 •  alter table cursos
4  drop column data;
```

- **AFTER** responsável por adicionar uma coluna em uma posição específica da tabela.

```
6 •  alter table cursos
7  add column idcurso int after idcurso;
```

Onde primeiramente adicionamos a coluna nova e em seguida o comando **AFTER** (que se traduzido temos a palavra após “Em seguida”), colocamos o nome da coluna existente que a nova coluna irá ser adicionado a frente.  
**OBS:** caso a coluna já exista e necessário apagar e adicionar novamente.

- **FIRST** responsável por adicionar uma coluna na primeira posição da tabela.

```
6 •  alter table cursos
7  add column idcurso int first;
```

- **MODIFY COLUMN** responsável por alterar o tipo da coluna.

```
3 •  alter table cursos
4  modify column nome varchar(50);
```

Neste caso mudamos o tipo da coluna para um varchar(50).

- **CHANGE COLUMN** responsável por alterar o nome da coluna.

```
6 •  alter table cursos
7  change column nome nome1 varchar(50);
```



**OBS:** Necessário especificar o tipo do campo ao alterar o nome como descrito no comando.

- **RENAME TO** responsável por alterar o nome da tabela.

```
9 • alter table cursos
10 rename to curso1;
```

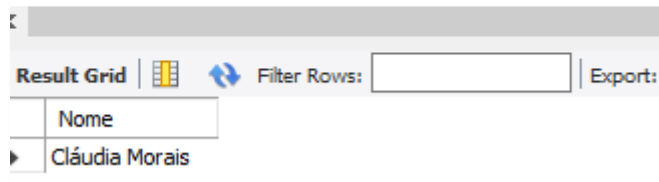
## Utilizando o SELECT para pesquisas.

- **Para que serve o comando SELECT?**

O comando SELECT basicamente e usado para realizar pesquisas de informações dentro do banco de dados, contendo vários complementos para uma pesquisa mais eficiente.

- **Where** Utilizando para adicionar uma condição para a pesquisa no comando **SELECT**.

```
21 • select Nome from tabela_de_vendedores
22 where nome = 'Cláudia Morais';
```



Nome
Cláudia Morais

Dessa forma será feito a listagem somente dos itens que tem como nome Cláudia Morais.

- **ORDER BY** Comando utilizado para organizar uma tabela por um elemento específico como Nome, CPF...

```
29 • select * from cursos
30 order by nome ;
31
32 • select * from cursos
33 order by nome desc;
34
35 • select nome, carga, ano from cursos
36 order by nome;
```



Podemos utilizar não somente o \* para selecionar toda a tabela, mas também somente os campos que queremos que sejam listados como no exemplo 03 da imagem a cima.

Temos o comando **DESC** que significa que a tabela irá ser organizada por ordem decrescente.

#### ➤ Filtrando linhas

- Temos algumas propriedades do **SQL** que podemos utilizar, alguns operadores lógicos como > (maior), < (menor), = (igual), >= (maior igual), <= (menor igual), AND (E)..
- Com esses operadores podemos montar uma quantidade imensa de formas de pesquisa para o SQL.

```
• select * from gafanhotos
  where nascimento >= '2000-01-01' and nascimento <= '2015-12-31'
  order by nascimento;

• select * from gafanhotos
  where profissao = 'programador';
```

#### Traduzindo o primeiro comando:

Selecione todos os itens de gafanhotos onde nascimento e maior ou igual a 01/01/2000 e nascimento menor ou igual a 31/12/2015, ordene por nascimento.

#### ➤ BETWEEN AND usando para organizar a tabela por dois elementos.

```
32 • select nome, ano from cursos
33   where ano between 2014 and 2016;

54 • select * from gafanhotos
55   where nascimento >= '2000-01-01' and nascimento <= '2015-12-31'
56   order by nascimento;
```

#### ➤ LIKE buscar elementos na tabela por letras

```
61 • select * from gafanhotos
62   where nacionalidade = 'Brasil' and nome like 'j%';
```

Nesse comando podemos por exemplo apresentar somente uma parte da informação que queremos buscar e automaticamente o SQL irá completar o resto.

Lembrando que e necessário colocar a % para que ele possa identificar.



- Temos algumas variações desse comando como '%p', '%p%', 'p\_'

➤ **DISTINCT** Mostrar quais categorias estão dentro de um campo.

```
1 • select distinct nacionalidade from gafanhotos;
```

nacionalidade
Brasil
Portugal
Moçambique
Irlanda
EUA
França
Japão
Canadá
Angola
Alemanha
Itália

Esse comando listara somente um item de cada. para ficar mais claro suponha que Japão está cadastrado três vezes na tabela, se utilizarmos esse comando ele só apareceria uma vez informando que esse item existe na estrutura.

➤ **COUT(\*)** Usado para saber quantos elementos a em uma determinada tabela ou especificação.

```
77 • select count(sexo) from gafanhotos  
78   where sexo = 'f' and altura > '1.90';
```

#### Explicando o comando:

Este comando ira me retornar uma contagem de quantas pessoas eu tenho dentro da tabela com o sexo f e altura maior que 1,90.

➤ **MAX** maior objeto dentro de um campo.

```
69 • select nome, max(altura), nacionalidade, sexo from gafanhotos  
70   where nacionalidade = 'brasil' and sexo = 'M';
```

➤ **MIN** menor objeto dentro de um campo selecionado.

```
74 • select sexo, min(peso) from gafanhotos  
75   where sexo = 'F' and nascimento >= '1990-01-01' and nascimento <= '2000-12-31';
```

➤ **SUM** somatória de todos os valores dentro do campo especificado.

```
52 • select sum(totaulas) from cursos;
```





- **AVG** Tirar a média de valores em um determinado campo.

```
54 • select avg(carga) from cursos;
```

## Agrupando e Agregando

- **GROUP BY** usado para agrupar elementos.

```
80 • select carga, count(nome) from cursos  
81   group by carga;
```

- **HAVING** Pegar mais de uma linha.

```
83 • select ano, count(*) from cursos  
84   where totaulas > 30  
85   group by ano  
86   having ano > 2013  
87   order by count(*) desc;
```

Mostrar somente a categoria que tiver mais de 3 cursos com um tanto de horas X.