



# Banco de Dados



# Criando um Banco de Dados No MySQL

# MySQL Server

Iremos utilizar o MySQL Server como ferramenta de estudo de banco de dados.



# XAMPP

Uma alternativa para não instalar o MySQL em seu computador está no XAMPP, que é um pacote com os principais servidores de código aberto do mercado.

Apache – **MySQL** – FileZilla – Mercury – TomCat



Download em [https://www.apachefriends.org/pt\\_br/index.html](https://www.apachefriends.org/pt_br/index.html)



# Entenda que...

A partir de agora iremos interagir com o MySQL através de comandos. A visualização de resultados não é muito clara.

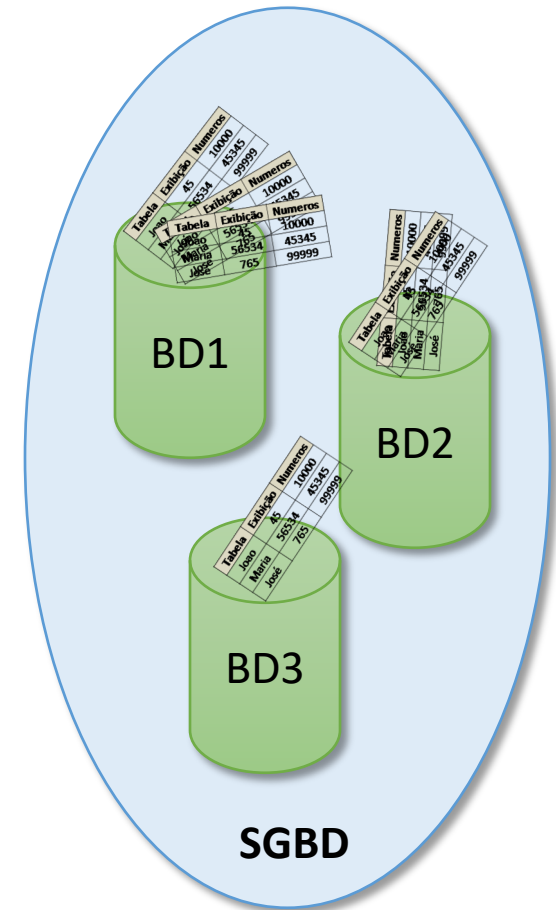
Então, antes de começarmos, lembre-se que:

Um SGBD é um programa que gerencia Banco de Dados (No caso, o MySQL).

Isso quer dizer que um SGBD pode gerenciar um ou mais (muitos) bancos de dados.

Um banco de dados possui tabelas com informações.

Logo, em um SGBD, podem existir diversos BDs, cada um com diversas tabelas, cada tabela com diversas informações.



# MySQL Command Line Client

Considerando que o MySQL Server já esteja instalado, e que você saiba a senha de administrador, vamos começar a trabalhar no MySQL.

Inicie o **MySQL Command Line Client**, da seguinte forma:

*Iniciar / Todos os Programas / MySQL / MySQL Server 5.x /  
MySQL Command Line Client*



# MySQL Command Line Client (Cont.)

Feito isso, será aberta uma janela preta, semelhante ao prompt do DOS, com a mensagem “Enter Password:”

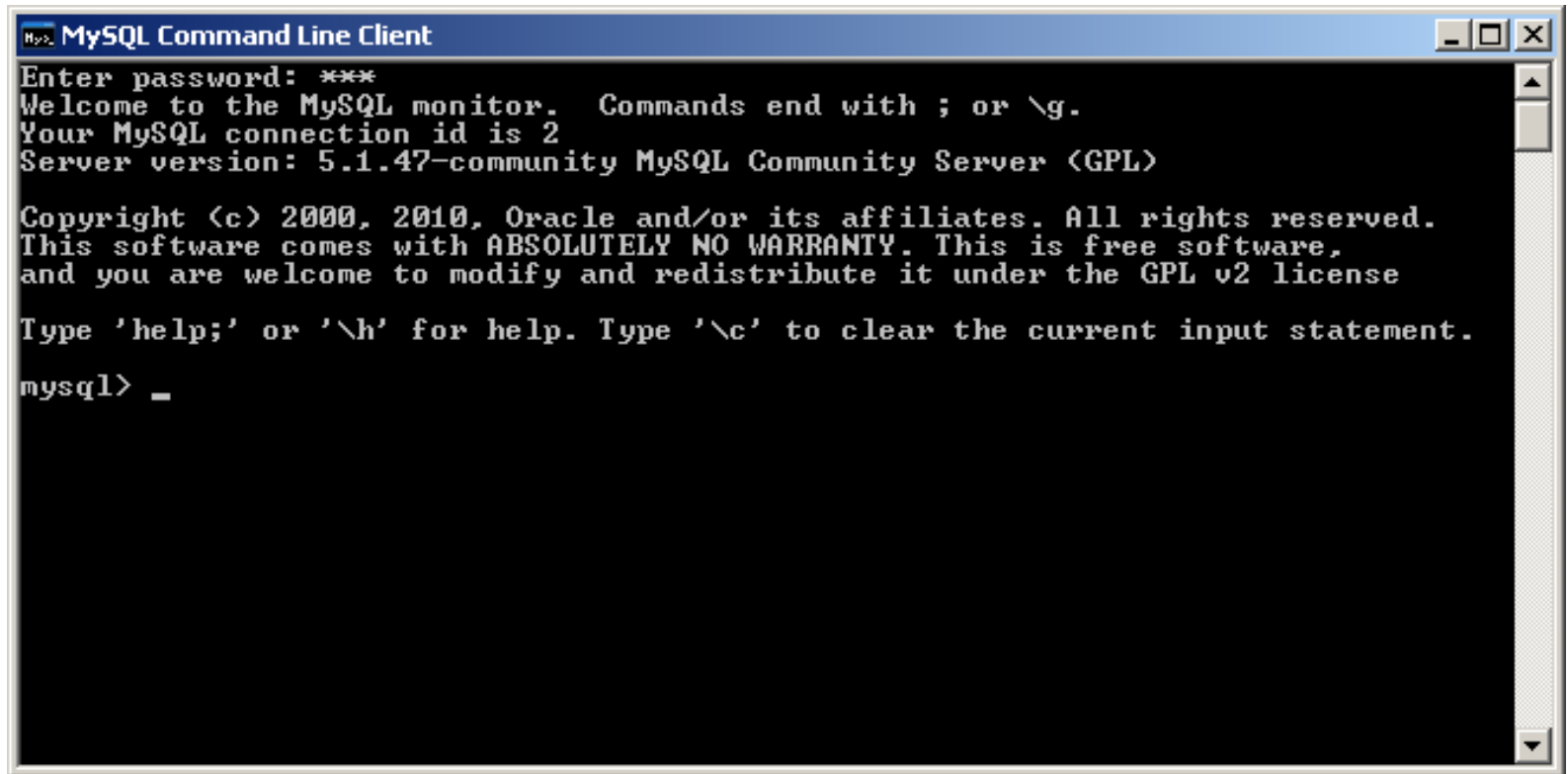
Digite a senha de Administrador que foi utilizada durante a instalação e pressione <Enter>. Este procedimento deverá ser feito sempre que o *MySQL Command Line Client* for executado.



# MySQL Command Line Client (Cont.)

Se você digitou corretamente a senha, o cursor agora estará pronto, aguardando novos comandos, da seguinte forma:

mysql> \_

A screenshot of the MySQL Command Line Client window. The title bar is blue and says "MySQL Command Line Client". The main area is black with white text. It shows the password prompt "Enter password: \*\*\*", a welcome message "Welcome to the MySQL monitor. Commands end with ; or \g.", connection details "Your MySQL connection id is 2" and "Server version: 5.1.47-community MySQL Community Server (GPL)", a copyright notice "Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved. This software comes with ABSOLUTELY NO WARRANTY. This is free software, and you are welcome to modify and redistribute it under the GPL v2 license", and usage instructions "Type 'help;' or '\h' for help. Type '\c' to clear the current input statement." The prompt "mysql> \_" is at the bottom.

```
MySQL Command Line Client
Enter password: ***
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.47-community MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> _
```

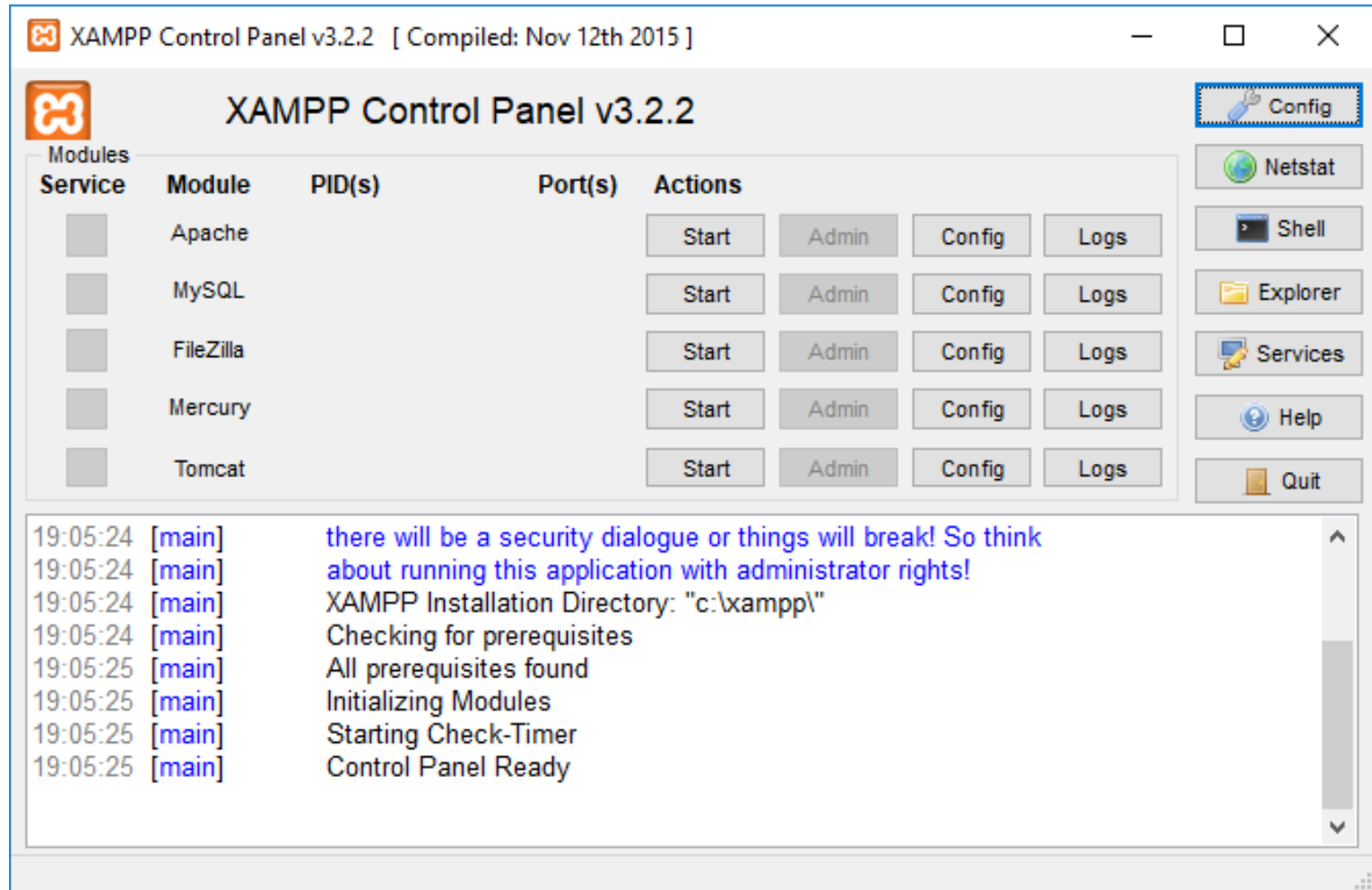




# Iniciando o MySQL pelo XAMPP

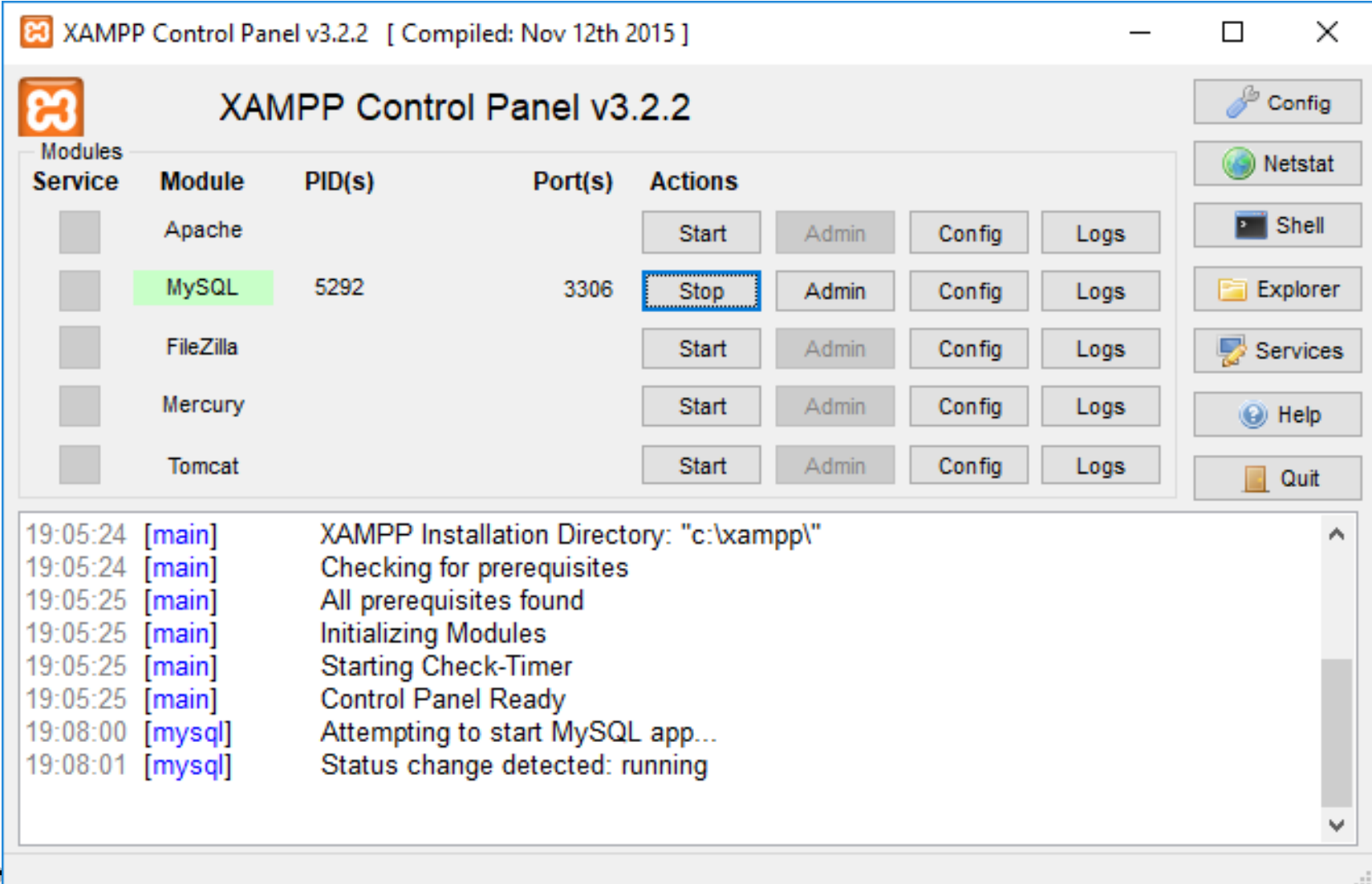
Se você preferiu utilizar o *MySQL* pelo *XAMPP*, faça o seguinte:

1 - Inicie o *XAMPP Control Panel*:



# Iniciando o MySQL pelo XAMPP

2 – Clique no botão *Start* referente ao *MySQL*:



XAMPP Control Panel v3.2.2 [ Compiled: Nov 12th 2015 ]

**XAMPP Control Panel v3.2.2**

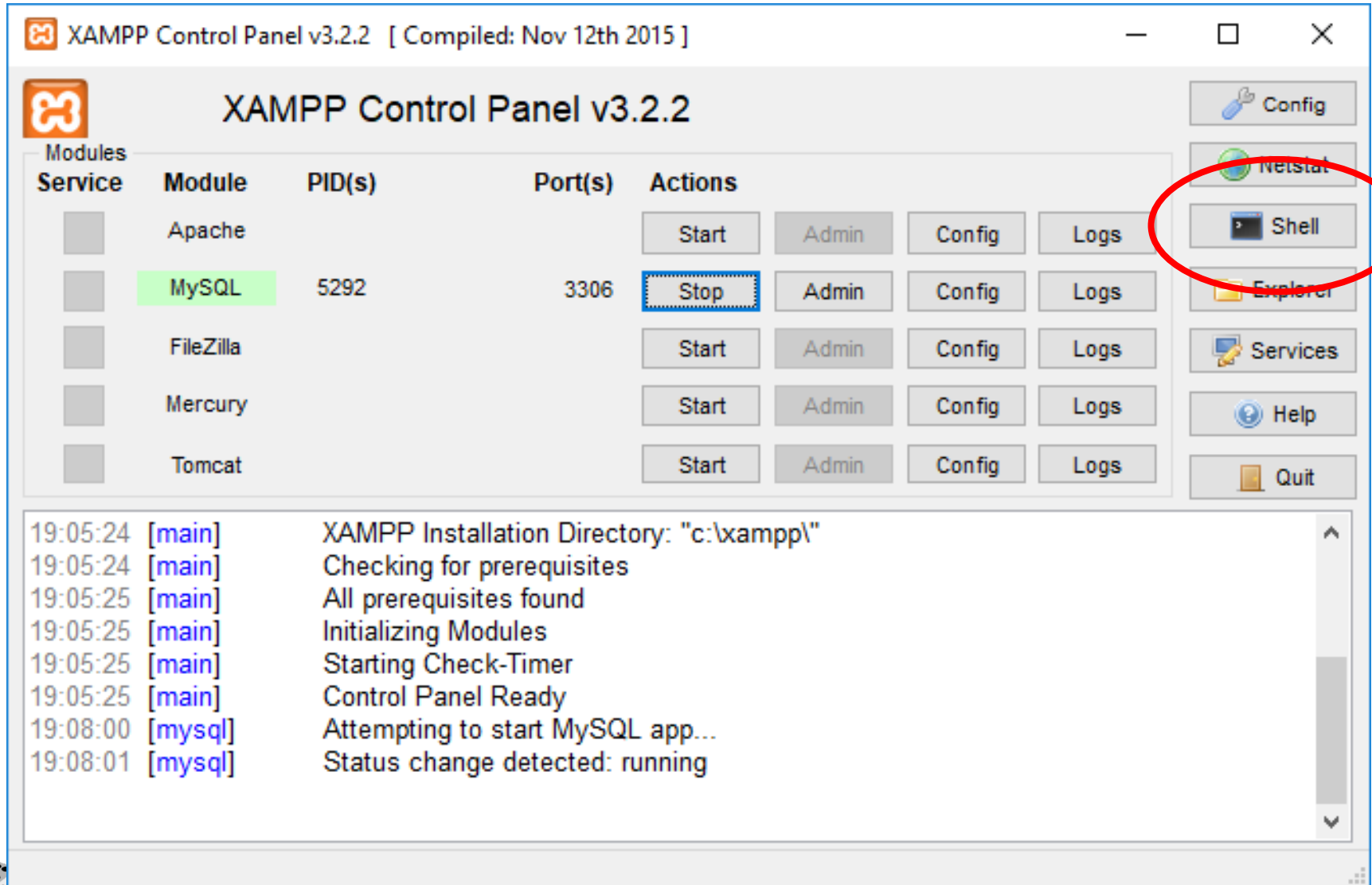
Service	Module	PID(s)	Port(s)	Actions
<input type="checkbox"/>	Apache			<input type="button" value="Start"/> <input type="button" value="Admin"/> <input type="button" value="Config"/> <input type="button" value="Logs"/>
<input type="checkbox"/>	MySQL	5292	3306	<input type="button" value="Stop"/> <input type="button" value="Admin"/> <input type="button" value="Config"/> <input type="button" value="Logs"/>
<input type="checkbox"/>	FileZilla			<input type="button" value="Start"/> <input type="button" value="Admin"/> <input type="button" value="Config"/> <input type="button" value="Logs"/>
<input type="checkbox"/>	Mercury			<input type="button" value="Start"/> <input type="button" value="Admin"/> <input type="button" value="Config"/> <input type="button" value="Logs"/>
<input type="checkbox"/>	Tomcat			<input type="button" value="Start"/> <input type="button" value="Admin"/> <input type="button" value="Config"/> <input type="button" value="Logs"/>

Config  
Netstat  
Shell  
Explorer  
Services  
Help  
Quit

19:05:24 [main] XAMPP Installation Directory: "c:\xampp\  
19:05:24 [main] Checking for prerequisites  
19:05:25 [main] All prerequisites found  
19:05:25 [main] Initializing Modules  
19:05:25 [main] Starting Check-Timer  
19:05:25 [main] Control Panel Ready  
19:08:00 [mysql] Attempting to start MySQL app...  
19:08:01 [mysql] Status change detected: running

# Iniciando o MySQL pelo XAMPP

3 – Inicie o modo texto pressionando o botão *Shell*:



XAMPP Control Panel v3.2.2 [ Compiled: Nov 12th 2015 ]

**XAMPP Control Panel v3.2.2**

Service	Module	PID(s)	Port(s)	Actions
<input type="checkbox"/>	Apache			Start Admin Config Logs
<input type="checkbox"/>	MySQL	5292	3306	Stop Admin Config Logs
<input type="checkbox"/>	FileZilla			Start Admin Config Logs
<input type="checkbox"/>	Mercury			Start Admin Config Logs
<input type="checkbox"/>	Tomcat			Start Admin Config Logs

Right-hand sidebar buttons: Config, Netstat, **Shell**, Explorer, Services, Help, Quit.

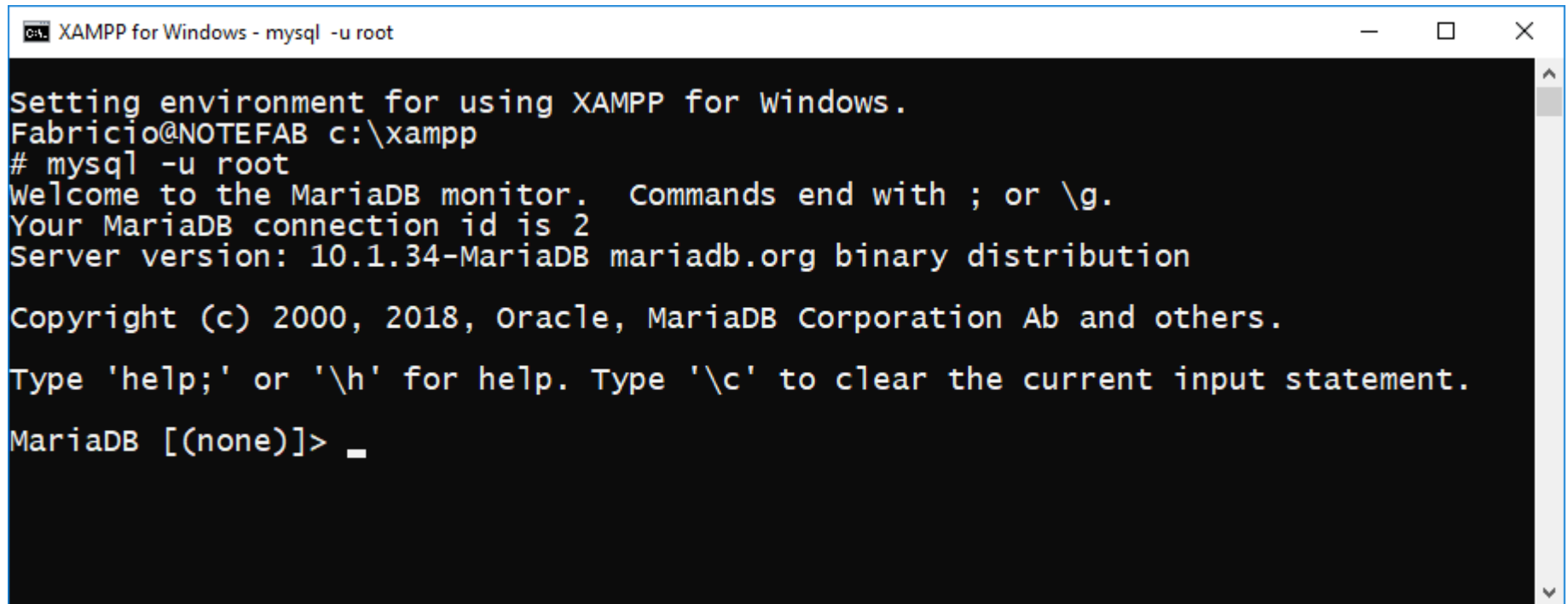
Log output:

```
19:05:24 [main] XAMPP Installation Directory: "c:\xampp\  
19:05:24 [main] Checking for prerequisites  
19:05:25 [main] All prerequisites found  
19:05:25 [main] Initializing Modules  
19:05:25 [main] Starting Check-Timer  
19:05:25 [main] Control Panel Ready  
19:08:00 [mysql] Attempting to start MySQL app...  
19:08:01 [mysql] Status change detected: running
```

# Iniciando o MySQL pelo XAMPP

4 – Digite o seguinte comando para iniciar o *Terminal MySQL*:

```
mysql -u root
```



```
C:\> XAMPP for Windows - mysql -u root

Setting environment for using XAMPP for windows.
Fabricio@NOTEFAB c:\xampp
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 10.1.34-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\\h' for help. Type '\\c' to clear the current input statement.

MariaDB [(none)]> _
```



# Visualizando os Bancos de Dados

Vamos então trabalhar com os primeiros comandos no prompt do MySQL.

Para visualizar os bancos de dados existentes no MySQL, utilize o comando a seguir:

**SHOW DATABASES ;**

Será exibido a lista de Bancos de Dados existentes.

Ex:

```
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| test |  
+-----+
```



# Criando um Banco de Dados

Para criar um Banco de Dados, utilize o seguinte comando:

```
CREATE DATABASE xxxxxx ;
```

xxxxxx é o nome que você dará ao BD. Não pode ter espaço nem caracteres especiais.



# Excluindo um Banco de Dados

Para excluir um Banco de Dados, utilize o seguinte comando:

```
DROP DATABASE xxxxx ;
```

xxxxx é o nome do BD que você quer excluir. É interessante visualizar os BDs existentes antes e depois de excluir algum.

**IMPORTANTE:** Toda a estrutura e dados contida no BD em questão será excluída de forma definitiva e irreversível. Por isso, certifique-se de estar excluindo o banco de dados que realmente deseja deletar.



# Ativando um Banco de Dados

Para ativar um BD, ou seja, para indicar ao MySQL com qual dos BDs existentes você pretende trabalhar, utilize o seguinte comando:

**USE xxxxxx ;**

xxxxxx é o nome do BD que você quer ativar. Obviamente que este BD precisa existir antes de ser ativado.





# Criando Uma Tabela

Para criar uma tabela, um BD precisa estar ativo.

Utilize o seguinte comando:

```
CREATE TABLE xxxxx (<colunas>) ;
```

xxxxx é o nome da tabela que você está criando. Este nome não deve ter espaços em branco, e deve ser inferior a 64 caracteres.

Aqui devem ser declarados os nomes de colunas da tabela, e cada domínio (tipo) de cada coluna, separados por vírgula. Ex:

```
CREATE TABLE ALUNOS (MATRICULA INT  
NOT NULL, NOME VARCHAR(64) NOT  
NULL, PRIMARY KEY (MATRICULA)) ;
```



# Principais Domínios (Tipos):

- **AUTO\_INCREMENT** – É usado para gerar um valor único sequencial para um novo registro. Só é possível ter uma coluna `auto_increment` na tabela, e esta coluna tem que ser chave primária. Um exemplo de criação de uma tabela com este recurso seria assim: *create table nometabela (nomecoluna int not null auto\_increment primary key, ...);*
- **BIGINT[(tamanho)] [UNSIGNED] [ZEROFILL]** – Utiliza-se esse tipo de dado quando usar valores inteiros grandes na faixa de -9.223.372.036.854.808 a 9.223.372.036.854.775.807. O parâmetro tamanho é opcional e estabelece o tamanho máximo do valor a ser exibido no monitor de vídeo, podendo ser um valor máximo 255. O parâmetro UNSIGNED, quando usado, estabelece que o valor definido será positivo, ou seja, podem ser utilizados valores na faixa 0 a 18.446.744.073.709.551.615. O parâmetro ZEROFILL estabelece a definição do preenchimento com valor zero caso o campo (coluna) seja deixado em branco.
- **CHAR(tamanho)** ou **CHARACTER(tamanho)** – Usado com sequências de caracteres de tamanho fixo que estejam limitadas a 255 caracteres de comprimento. O parâmetro tamanho determina o valor máximo em caracteres que pode conter a sequência. Esse tipo de dado, quando definido, preenche o campo com espaços em branco até completar o total de caracteres definidos, quando a totalidade do tamanho do campo não é preenchida.
- **DATE** – Utilizado com uma data de calendário no formato AAAA-MM-DD (Formato ANSI) dentro do intervalo de tempo entre 1000-01-01 e 9999-12-31.
- **DECIMAL(tamanho, decimal)** [UNSIGNED] [ZEROFILL] – Quando for preciso usar valores com ponto flutuante como se fossem uma sequência de caracteres do tipo CHAR. O parâmetro decimal determina o número de casas decimais. O parâmetro UNSIGNED, quando usado, estabelece que o valor definido será positivo. O parâmetro ZEROFILL estabelece a definição do preenchimento com valor zero caso o campo (coluna) seja deixado em branco.
- **DOUBLE(tamanho, decimal)** [UNSIGNED] [ZEROFILL] – Quando for necessário usar valores com tamanho normal (dupla precisão) na faixa de valores -1.7976931348623157E+308 e -2.2250738585072014E-308, 0 e entre 2.2250738585072014E-308 e 1.7976931348623157E+308. O parâmetro decimal determina o número de casas decimais. O parâmetro UNSIGNED, quando usado, estabelece que o valor definido será positivo. O parâmetro ZEROFILL estabelece a definição do preenchimento com valor zero caso o campo (coluna) seja deixado em branco.

# Principais Domínios (Tipos): (Cont.)

- **ENUM** (`'valor1'`, `'valor2'`, ... , `'valorN'`) – Usado com uma lista de valores do tipo string, em que um dos valores pode ser selecionado. O campo (coluna) do tipo ENUM pode possuir até no máximo 65535 valores diferentes.
- **FLOAT**[(`tamanho`, `decimal`)] [**UNSIGNED**] [**ZEROFILL**] – Quando usar valores com ponto flutuante pequenos (precisão simples) na faixa de valores -3.402823466E+38 a -1.175494351E-38, 0, e entre 1.175494351E-38 e 3.402823466E+38. O parâmetro `tamanho` é opcional e permite estabelecer o tamanho máximo a ser exibido no monitor de vídeo, podendo ser um valor máximo 255. O parâmetro `decimal` determina o número de casas decimais. O parâmetro **UNSIGNED**, quando usado, estabelece que o valor definido será positivo. O parâmetro **ZEROFILL** estabelece a definição do preenchimento com valor zero caso o campo (coluna) seja deixado em branco.
- **INTEGER**[(`tamanho`)] [**UNSIGNED**] [**ZEROFILL**] ou **INT**[(`tamanho`)] [**UNSIGNED**] [**ZEROFILL**] - utilizado quando houver a necessidade de usar valores inteiros longos, na faixa de -2.147.483.648 até 2.147.483.647. O parâmetro `tamanho` é opcional e permite estabelecer o tamanho máximo a ser exibido no monitor de vídeo, podendo ser um valor máximo 255. O parâmetro **UNSIGNED**, quando usado, estabelece que o valor definido será positivo. O parâmetro **ZEROFILL** estabelece a definição do preenchimento com valor zero caso o campo (coluna) seja deixado em branco.
- **SMALLINT**[(`tamanho`)] [**UNSIGNED**] [**ZEROFILL**] – Quando for necessário usar valores inteiros curtos na faixa de -32.768 até 32.767. O parâmetro `tamanho` é opcional e permite estabelecer o tamanho máximo a ser exibido no monitor de vídeo, podendo ser um valor máximo 255. O parâmetro **UNSIGNED**, quando usado, estabelece que o valor definido será positivo. O parâmetro **ZEROFILL** estabelece a definição do preenchimento com valor zero caso o campo (coluna) seja deixado em branco.
- **TIME** – Utiliza-se esse tipo de dado quando houver necessidade de usar informação relacionada a um determinado horário de relógio no intervalo de tempo entre `'-838:59:59'` e `'838:59:59'`. Os valores de hora do MySQL são mostrados no formato `'HH:MM:SS'`, no entanto, é possível atribuir valores de colunas (campos) usando strings ou números.



# Principais Domínios (Tipos): (Cont.)

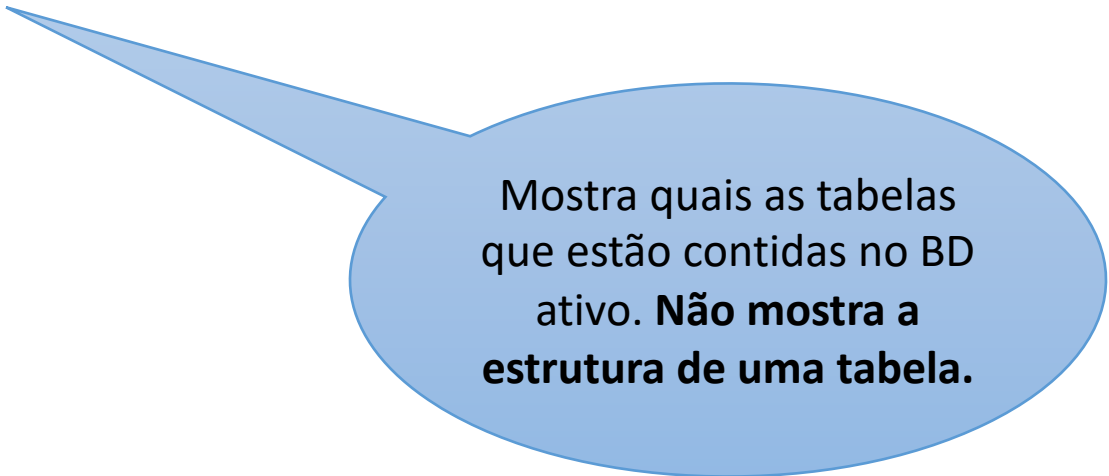
- **TINYINT[(tamanho)] [UNSIGNED] [ZEROFILL]** – Quando precisar de valores inteiros pequenos na faixa de -128 até 127. O parâmetro tamanho é opcional e permite estabelecer o tamanho máximo a ser exibido no monitor de vídeo, podendo ser um valor máximo 255. O parâmetro UNSIGNED, quando usado, estabelece que o valor definido será positivo. O parâmetro ZEROFILL estabelece a definição do preenchimento com valor zero caso o campo (coluna) seja deixado em branco.
- **VARCHAR(tamanho)** – Quando precisa de sequências de caracteres de tamanho variável que estejam limitadas a 255 caracteres de comprimento. A diferença entre esse tipo e o CHAR é que, neste caso, os espaços em branco excedentes do lado direito da sequência de caracteres não utilizados são automaticamente desprezados. O parâmetro tamanho determina o valor máximo da sequência em caracteres.



# Visualizando Tabelas

Se você fez os passos anteriores sem nenhum erro, utilize agora o seguinte comando para visualizar todas as tabelas contidas no BD ativo:

**SHOW TABLES;**



Mostra quais as tabelas que estão contidas no BD ativo. **Não mostra a estrutura de uma tabela.**



# Visualizando A Estrutura De Uma Tabela

A sintaxe é a seguinte:

```
DESCRIBE xxxxx [coluna] ;
```

xxxxx é o  
nome da  
tabela

Aqui é opcional, caso a intenção seja  
visualizar apenas uma coluna. Deve  
ser indicado o nome da coluna, sem  
os colchetes.



# Excluindo Uma Tabela

Para excluir uma tabela, um BD precisa estar ativo.

Utilize o seguinte comando:

```
DROP TABLE xxxxxx ;
```

xxxxxx é o nome da tabela que  
você está excluindo.

**IMPORTANTE:** A tabela em questão será excluída de forma definitiva e irreversível. Por isso, certifique-se de estar excluindo a tabela que realmente deseja deletar. Se esta tabela já contiver registros inseridos, estes também serão perdidos.



# Exemplo para Teste de Aprendizagem:

- 1 - Crie e ative um BD chamado ***empresa***.
- 2 - Crie uma tabela chamada ***cliente***, com as seguintes características:

Field	Type	Null	Key	Default	Extra
matricula	int(11)	NO	PRI	NULL	auto_increment
nome	varchar(60)	NO		NULL	
telefone	char(13)	YES		NULL	
email	varchar(30)	YES		NULL	
nascimento	date	NO		NULL	

- 3 – Avalie a se a estrutura da tabela ficou exatamente igual ao exemplo acima, pelo comando ***describe cliente;***
- 4 – Se não ficou exatamente igual, apague a tabela e refaça.





# Exemplo para Teste (Cont.):

Field	Type	Null	Key	Default	Extra
matricula	int(11)	NO	PRI	NULL	auto_increment
nome	varchar(60)	NO		NULL	
telefone	char(13)	YES		NULL	
email	varchar(30)	YES		NULL	
nascimento	date	NO		NULL	

O comando correto para criar a tabela conforme o modelo é:

***create table cliente***

***(matricula int primary key auto\_increment,***

***nome varchar(60) not null,***

***telefone char(13),***

***email varchar(30),***

***nascimento date not null);***



# Inserindo Registros Em Tabela

A partir do momento em que uma tabela está pronta, ela já pode receber a entrada de dados, que podem ser enviados de duas formas:

1 - Forma direta – Instrução **INSERT INTO**

2 - Forma indireta – Instrução **LOAD DATA**



# 1 - Forma direta – INSERT INTO

Sintaxe para a instrução **INSERT INTO**:

```
INSERT INTO <tabela>
```

```
(Campo1,  
 Campo2,  
 CampoN)
```

```
VALUES
```

```
Valor1,  
Valor2,  
ValorN);
```

Repita esta instrução, alterando os dados, para catalogar outros clientes nesta tabela.

Exemplo para a tabela que estamos trabalhando:

```
INSERT INTO cliente  
  (matricula,  
   nome,  
   telefone,  
   email,  
   nascimento)  
VALUES  
  (null,  
   'Ayrton Bueno',  
   '(21)8511-9876',  
   'abueno@email.com',  
   '1990-10-25');
```

opcional



# 1 - Forma direta – INSERT INTO (Cont.)

Exercício:

Insira agora na tabela Cliente, 10 registros diferentes pela forma direta, sendo que:

- 3 pessoas possuem exatamente o mesmo número de telefone
- Outras 2 pessoas possuem a mesma data de nascimento
- Outras 2 pessoas possuem exatamente o mesmo nome

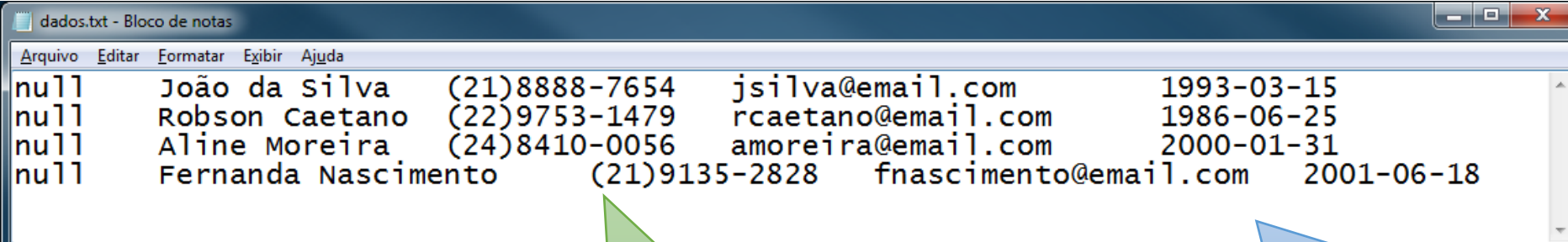


## 2 - Forma indireta – LOAD DATA

A instrução **LOAD DATA** tenta facilitar a inserção de dados em tabelas. Para isso faz-se necessário a utilização do **Bloco de Notas** (ou outro editor de texto puro).

Abra o bloco de notas e insira os dados formando as colunas da tabela. **Cada coluna fica separada da outra usando-se a tecla <tab> do teclado.**

Exemplo de acordo com a tabela que estamos trabalhando:



```
dados.txt - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda
null    João da Silva    (21)8888-7654    jsilva@email.com    1993-03-15
null    Robson Caetano  (22)9753-1479    rcaetano@email.com   1986-06-25
null    Aline Moreira   (24)8410-0056    amoreira@email.com   2000-01-31
null    Fernanda Nascimento (21)9135-2828    fnascimento@email.com 2001-06-18
```

Salve o arquivo com a extensão .txt, dentro da pasta *bin* do MySQL

Não se importe com a aparência do texto, e sim, com a **separação de dados APENAS pela tecla <tab>**



## 2. Forma indireta – LOAD DATA (Cont.)

Com o arquivo txt salvo dentro da pasta bin do MySQL, insira então a seguinte instrução:

```
LOAD DATA LOCAL INFILE "dados.txt" INTO TABLE cliente;
```

Nome do arquivo txt  
que você criou com os  
dados

Nome da tabela onde  
os dados do arquivo txt  
serão inseridos



# OBSERVAÇÃO

Como foi utilizado nos dois slides anteriores, se uma tabela possui um campo (coluna) que seja de auto incremento, você precisará, no momento de inserir dados nesta tabela, informar para este campo o valor **null**.

Mais um exemplo:

Se você criou uma tabela assim:

```
CREATE TABLE cadastro (CODIGO INT(5) NOT NULL  
AUTO INCREMENT PRIMARY KEY, NOME VARCHAR(60) ,  
ENDERECO VARCHAR(120)) ;
```

Terá que inserir uma linha de registro assim:

```
INSERT INTO cadastro VALUES (NULL, 'FULANO DE  
TAL' , 'RUA SOBE E DESCE, SEM NÚMERO' ) ;
```

Ou preencher no bloco de notas com o termo **NULL** no campo referente a CODIGO.



## 2. Forma indireta – LOAD DATA (Cont.)

Exercício:

Insira agora na tabela Cliente, 5 registros diferentes pela forma indireta, sendo que:

- 2 pessoas possuem exatamente o mesmo nome.





# Consulta de Registros

O comando **SELECT** possui diversos parâmetros de utilização, e é no conhecimento destes parâmetros que está baseada boa parte de nossos estudos em MySQL.

Vamos começar pelo comando **SELECT** mais básico de todos.

Visualize todo o conteúdo da tabela *cliente* com o seguinte comando:

```
SELECT * FROM cliente;
```



# Consulta de Registros (Cont.)

Para extrair listagens mais específicas de registros armazenados em uma tabela, utilize o comando **SELECT**, anexado a alguns parâmetros:

[tipo] é opcional, podendo ser:

- **DISTINCT** (registros distintos)
- **ALL** (todos os registros)

<tabela> é o nome da tabela (ou tabelas) de onde se deseja consultar os registros.

**SELECT** [tipo] <campos> **FROM** <tabela> [condição];

<campos> é a lista de campos a serem selecionados. Pode ser utilizado o valor **\*** (asterisco) para todos os campos da tabela

[condição] é opcional e determina a condição de ação de pesquisa, podendo ser:

WHERE		Determina a ação de trabalho de uma condição baseada em uma relação lógica.
GROUP BY	ASC	Indica o agrupamento de informações baseado em valores comuns a partir de uma coluna informada.
	DESC	
ORDER BY	ASC	Define a forma de ordenação dos registros, podendo ser ASC (ascendente) e DESC (descendente)
	DESC	



# Consulta de Registros (Cont.)

Tente agora estas outras opções de consulta sugeridas abaixo, e **compare os resultados** no intuito de entendimento do respectivo código inserido:

```
SELECT * FROM cliente;
```

```
SELECT NOME, TELEFONE FROM cliente;
```

```
SELECT TELEFONE, NOME FROM cliente;
```

```
SELECT NOME, EMAIL FROM cliente WHERE MATRICULA = 5;
```

```
SELECT NOME, NASCIMENTO FROM cliente ORDER BY NOME;
```

```
SELECT NOME, MATRICULA, TELEFONE FROM cliente ORDER BY  
NOME DESC;
```



# Consulta de Registros (Cont.)

```
SELECT NOME, NASCIMENTO FROM cliente ORDER BY  
NASCIMENTO, NOME DESC;
```

```
SELECT NOME FROM cliente WHERE NASCIMENTO = 'ddd' ORDER  
BY NOME;
```

*Substituir **ddd** pela data repetida que foi inserida.*

```
SELECT NOME, TELEFONE FROM cliente WHERE NOME = 'nnn'  
ORDER BY MATRICULA DESC;
```

*Substituir **nnn** pelo nome  
repetido que foi inserido.*

```
SELECT DISTINCT NOME FROM cliente;
```

```
SELECT DISTINCT NOME, TELEFONE FROM cliente;
```

```
SELECT MATRICULA, NOME FROM CLIENTE WHERE TELEFONE =  
'ttt' ORDER BY NASCIMENTO;
```

*Substituir **ttt** pelo telefone  
repetido que foi inserido.*



# Consulta de Registros (Cont.)

A título de verificação de funcionalidade de campos com domínio **date**, será apresentada a relação de todos os clientes nascidos no mês de dezembro de qualquer ano, se for utilizada a seguinte sintaxe:

```
SELECT NOME, NASCIMENTO FROM cliente WHERE  
MONTH(NASCIMENTO) = 12;
```

A seguir observe a listagem de todos os clientes nascidos a partir de 1º de janeiro de 2000. Para tanto, use a seguinte sintaxe:

```
SELECT NOME, NASCIMENTO FROM cliente WHERE  
NASCIMENTO >= '2000-01-01';
```



# Alteração de Registros

O processo de alteração de registros é realizado no MySQL pelo comando abaixo:

```
UPDATE <tabela> SET <campo> = <expressão> [condição];
```

Nome da a tabela que  
terá registros alterados  
/ atualizados

É a indicação de um  
campo da tabela

É a indicação do  
valor do campo a  
ser atualizado

É opcional. Determina a condição de  
ação de pesquisa baseada no  
argumento **WHERE**



# Alteração de Registros (Cont.)

Vamos então aplicar um comando de alteração de registros em nosso BD de estudo.

Primeiro faça uma consulta ao(s) registro(s) que será(ão) alterado(s) posteriormente:

```
SELECT * FROM cliente;
```

Observe bem os clientes que possuem telefone repetido nesta consulta. Agora vamos aplicar a alteração:

```
UPDATE cliente SET TELEFONE = '(21)1111-5555'  
WHERE MATRICULA = m;
```

Substitua **m** pela matrícula de um dos clientes com telefone repetido

Agora repita o comando **SELECT** acima e observe os telefones.



# Alteração de Registros (Cont.)

Agora troque o nome de uma pessoa que possui nome repetido em sua tabela cliente. Como deve ficar este comando?

Visualização inicial:

```
SELECT * FROM cliente;
```

Alteração:

```
UPDATE cliente SET NOME = 'nnn' WHERE MATRICULA = m;
```

**nnn** = um novo nome

**m** = matricula de um dos nomes repetidos

Visualização final:

*Basta repetir o comando da visualização inicial e observar agora na coluna nome que o antigo nome repetido foi substituído pelo recente nome alterado.*





# Remoção de Registros

Vamos estudar o comando para remover registros:

**DELETE FROM <tabela> [condição] ;**

Nome da a tabela que  
terá registros removidos

É opcional. Determina a ação  
da pesquisa baseada no  
argumento **WHERE** .

**Porém, se não houver  
uma condição, todo o  
conteúdo da tabela  
será removido.**



# Remoção de Registros (Cont.)

Vamos então ao treino (com cautelas, para não perdermos dados por descuido):

Visualize todos os registros da tabela cliente. Creio que não seja mais preciso apresentar o comando para isso, ok?

Agora exclua um conjunto de registros, com este comando:

```
DELETE FROM cliente WHERE NASCIMENTO = 'nnn' ;
```

Substituir **nnn** pela data de nascimento que está repetida

Visualize novamente todos os registros e observe as alterações.



# Alteração de Tabelas

É possível **alterar a estrutura de uma tabela**. Para isso, segue o comando do MySQL:

Nome da a tabela que terá a alteração de estrutura.

**ALTER TABLE <tabela> <operação> [complemento];**

- **ADD <campo> <tipo> [after nomecoluna | first]**

Adiciona um campo (coluna) à tabela. Opcionalmente, caso queira a nova coluna após uma determinada coluna existente, adicionar o termo **after** seguido do nome da coluna existente. Caso queira que a nova coluna seja a primeira da tabela, adicionar o termo **first**. (After e First são opcionais.)

- **DROP <campo>**

Apaga um campo (coluna) da tabela.

- **MODIFY <campo> <novo tipo>**

Modifica o tipo de uma coluna (Ex: de INT para SMALLINT).

- **RENAME TO <novo nome da tabela>**

Renomeia a tabela.

- **CHANGE <nome do campo> <novo nome do campo> <novo tipo>**

Renomeia o campo (coluna), sendo necessário incluir o tipo deste campo que está sendo renomeado, que pode ser o mesmo tipo anterior, ou pode ser alterado.



# Alteração de Tabelas (Cont.)

Vamos treinar, inserindo um campo em nossa tabela de estudos:

```
ALTER TABLE cliente ADD CPF CHAR(14) ;
```

Agora visualize todo o conteúdo da tabela **cliente**.

Observe que a coluna **CPF** é indicada com valores “**NULL**”, o que indica que está sem registros.

*Agora insira os dados na nova coluna CPF. Veja um exemplo para sua tarefa:*

```
UPDATE cliente SET CPF = '031.285.342-05' WHERE MATRICULA = 6;
```

*Faça isso para todos os clientes cadastrados na tabela, colocando um CPF diferente para cada um.*



# Exercício

Inclua os registros de mais 5 clientes na tabela *cliente*, preenchendo todos os campos de registro, sem repetição de dados entre estes novos registros. Escolha o método de inserção de registros que mais lhe agrada.



# Dúvidas?

