

MARCOS DE MELO

Banco de dados MySQL

1ª Edição

Banco de dados MySQL

Sumário

Aula 1 – Introdução	6
Surgimento do MySQL.....	6
Licença de uso	6
Para que usar um banco de dados?.....	6
SGDB.....	7
Conceito de banco de dados relacional	7
Campos	7
Registros.....	8
Tabelas	8
Definição de nomes de Banco de dados, Tabelas, Índice, Coluna e Alias	8
Tipos de dados.....	9
Tipos de dados Numéricos.....	9
Tipos de dados Data/Hora	10
Tipos de dados String.....	10
Conceitos básicos sobre banco de dados relacional	11
Tabelas relacional	11
Aula 2 - Instalação do MySQL	13
O que é o XAMPP?.....	13
Instalando o pacote de programas XAMPP	13
Acessando o banco de dados MySQL pela primeira vez	18
Criando seu primeiro banco de dados	22
Criando a primeira tabela	24
Inserindo dados na tabela.....	25
Visualizando os registros cadastros nas tabelas	27
Atividades.....	29

Aula 3 – Criação de Bancos e Tabelas	30
O Workbench (GUI tool)	30
Instalando e usando o Workbench (GUI tool)	30
Executando o Programa Workbench	32
Interface do programa	32
Criando uma nova instância de conexão	34
Criando banco de dados e suas tabelas	36
Criando um novo banco de dados.....	36
Visualizando bancos de dados	36
Ativando um banco de dados	37
Deletando um banco de dados	37
Criando tabelas.....	37
Exibindo as tabelas existentes no banco de dados	39
Exibir descrições de uma tabela	39
Adicionando um campo em uma tabela existente	40
Alterando um campo em uma tabela existente	40
Deletando uma coluna em uma tabela existente	41
Deletando uma tabela existente	41
Atividades.....	43
Aula 4 – Instrução Select	45
Tabelas de exemplos para consultas	45
Comando básico do select	47
Atividades.....	49
Aula 5 - Cláusula Where / Operadores de Comparação	50
Operadores de comparação	50
Igual a “ = “	50
Diferente “ != “ ou “<>”	51
Maior que “ > ”	51

Maior ou igual a “ >= ”	52
Menor que “ < ”	52
Menor ou igual a “ <= ”	53
Atividades	54
Aula 6 – Cláusula Where / Comandos Especiais SQL	55
Operadores SQL especiais	55
BETWEEN	55
IN (val1, val2, val3, val4)	56
LIKE	56
IS NULL	57
Atividades	58
Aula 7 –Parâmetros da instrução Select / Parte 1	59
Parâmetro JOIN	59
Parâmetro ORDER BY	60
Parâmetro DISTINCT(Omite registros duplicados).....	60
Parâmetro LIMIT (limitador de registros selecionados).....	61
Parâmetro COUNT (Contagem de registros).....	62
Aula 8 –Parâmetros da instrução Select / Parte 2	64
Parâmetro SUM (Soma de valores)	64
Parâmetro AVG (Média de valores).....	64
Parâmetro MAX/MIN	65
Parâmetro GROUP BY.....	66
Parâmetro HAVING.....	67
Aula 9 – Instruções de Inserções, atualizações e exclusões	68
Insert	68
Update.....	68
Delete	70

Atividades.....	71
Aula 10 – Controle de Usuários / Backups e Recovers	72
Controle de Usuários	72
Criando um usuário	72
Exibindo usuários cadastrados	73
Excluindo um usuário	73
Liberando Privilégios.....	74
Backups e Recovers	76
Backup pelo phpMyAdmin	76
Criando um backup.....	78
Restaurando um Backup.....	80
Aula 11 – Revisão.....	83
Exercício 1 – Criando um banco de dados e suas tabelas	83
Exercício 2 - Questionário.....	84
Aula 12 – Avaliação.....	88

Aula 1 – Introdução

Seja bem-vindo ao curso de desenvolvimento de banco de dados em MySQL, caro aluno e amigo. Este livro tem como objetivo abordar a instalação, entendimento e utilização do banco de dados open source mais utilizado no mundo, o MySQL. O aluno aprenderá os principais comandos utilizados em banco de dados em sistemas Web.

A compreensão do conceito de banco de dados e das tecnologias relacionadas com a sua utilização, são fundamentais para os profissionais de desenvolvimento de sistemas Web, principalmente porque o banco de dados é a base de um sistema Web.

Surgimento do MySQL

Foi desenvolvido no ano de 1980 na Suécia por dois suecos e um finlandês; David Axmark, Allan Larsson e Michael "Monty" Wideniuse. Com o sucesso do MySQL, criaram a empresa MySQLAB que difundiu então o uso do MySQL pelo mundo a fora.

No dia 16 de Janeiro de 2008, a MySQL AB, foi comprada pela Sun Microsystems, por US\$ 1 bilhão de dólares. No dia 20 de Abril de 2009, foi anunciado que a Oracle compraria a Sun Microsystems e todos os seus produtos, incluindo o MySQL.

Licença de uso

Atualmente a Oracle, atual proprietária do MySQL continua disponibilizando o MySQL gratuitamente, sendo um Software Livre com base na GPL (Licença Pública Geral) mas, se o programa que acessar o Mysql não for GPL, uma licença comercial deverá ser adquirida.

Para que usar um banco de dados?

É muito fácil dar exemplos da usabilidade de sistemas de banco de dados nas aplicações web. Como exemplo podemos citar, instituições financeiras (bancos), lojas online, sistemas web em geral, são exemplos claros desta necessidade. Imagine a seguinte situação: Um usuário acessa um site comercial de venda online de eletrônicos, com uma variedade imensa de produtos disponíveis para

compra. As informações sobre cada produto, como, código do produto, descrição, valor, entre outros, são considerados campos de um registro, que são dados armazenados em um **sistema gerenciador de banco de dados** ou simplesmente **SGDB**. Para realizar a compra do produto, geralmente o sistema solicitara dados do cliente, como, nome, cpf, endereço, e-mail e muito mais. Estes dados serão salvos no banco de dados da loja virtual, para que o cliente não precise em uma nova compra, seus dados novamente.

SGDB

Sistemas de Gerenciamento de Banco de dados são a maneira mais eficaz de armazenar e pesquisar dados relacionais, possibilitando aos usuários utilizarem uma grande variedade de abordagens no tratamento das informações. Todos os bancos de dados relacionais atualmente em uso no mundo, são manipulados pela linguagem **SQL (Structured Query Language)**, ou **Linguagem de Consulta Estruturada** em português.

A linguagem SQL foi criada originalmente pela IBM no início dos anos 70 e hoje é um padrão para todos os bancos de dados relacionais. Esse padrão da linguagem foi determinado pela **American National Standards Institute (ANSI)** em 1986 e ISO em 1987.

Conceito de banco de dados relacional

Um **Banco de Dados Relacional** é um conceito abstrato que define maneiras de armazenar, manipular e recuperar dados estruturados, modelados unicamente como dados em tabelas, originando um banco de dados. Explicando de maneira resumida, podemos dizer que, um banco de dados é um local onde armazenamos informações para posteriormente realizarmos consultas a estas informações distintas, agrupadas em forma de registros no formato de tabelas.

O banco de dados é constituído por três elementos principais: **campos**, **registros** e **tabelas**.

Campos

É exatamente o local onde colocamos determinada informação. Este local é normalmente nomeado com palavras que caracteriza a informação armazenada dentro dele, por exemplo, um campo nomeado como “**Endereço**” guardaria obviamente, informações relacionadas a esta palavra.

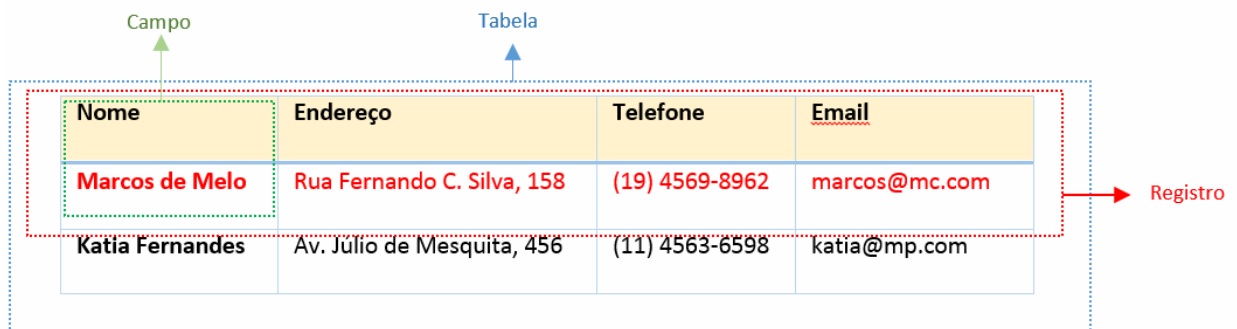
Registros

Registros é um conjunto de campos com informações normalmente relacionadas a um determinado assunto. Por exemplo, vamos supor que o assunto em questão seja “dados pessoais”, o conjunto de campos relacionadas poderia ser então nome, endereço, fone. Cada conjunto de informações alimentado por informações, constitui um novo registro de dados cadastrado.

Tabelas

Tabela de banco de dados armazenam um determinada grupo de registros por categoria, ou seja, uma tabela é uma categoria que armazena um conjunto de registros de um determinado assunto.

Abaixo podemos entender a estrutura de uma tabela de dados;



Estrutura de uma tabela, seus registros e campos

Definição de nomes de Banco de dados, Tabelas, Índice, Coluna e Alias

Identificador	Tamanho máximo	Caracteres permitidos
Banco de dados	64	Qualquer caractere que é permitido em um nome de diretório exceto "/" ou ".".
Tabela	64	Qualquer caractere permitido em um nome de arquivo, exceto
Coluna	64	Todos os caracteres
Alias	255	Todos os caracteres

Tipos de dados

Ao criar uma tabela você deverá especificar o tipo de dados a ser armazenado em cada campo, nela especificado. Para cada campo de cada uma das tabelas, é necessário determinar o tipo de dados que ele poderá armazenar, e conseguir um armazenamento com a menor utilização de espaço possível.

O MySQL possui três tipos de dados básicos de tipos de dados: Numéricos, Data/Hora e String.

Tipos de dados Numéricos

TIPO	INTERVALO	BYTES	DESCRIÇÃO
TINYINT[(M)]	-127 a 128; ou 0 a 255	1	Inteiros muito pequenos
BIT			O mesmo que TINYINT
BOOL			O mesmo que TINYINT
SMALLINT[(M)]	-32768 a 32767	2	Inteiros pequenos
MEDIUMINT[(M)]	-8388608 a 8388607; ou 0 a 16777215	3	Inteiros de tamanho médio
INT[(M)]	-213 a 231-1; ou 0 a 232-1	4	Inteiros regulares
INTEGER[(M)]			O mesmo que INT
BIGINT[(M)]	-2 ⁶³ a 2 ⁶³ -1; ou 0 a 2 ⁶⁴ -1	8	Inteiros grandes
FLOAT(precisão)	Depende da precisão	Variável	Números de ponto flutuante de precisão simples ou dupla
FLOAT[(M,D)]	1.175494351E-38 a ±3.402823466E+38	4	Números de ponto flutuante de precisão simples. O mesmo que FLOAT(4)
DOUBLE[(M,D)]	±1.7976931348623157E+308 a ±2.2250738585072014E-308	8	Números de ponto flutuante de precisão dupla. O mesmo que FLOAT(8)
DOUBLE			O mesmo que DOUBLE[(M,D)]
PRECISION[(M,D)]			O mesmo que DOUBLE[(M,D)]
REAL[(M,D)]			O mesmo que DOUBLE[(M,D)]
DECIMAL[(M,D)]	Variável	M+2	Número de ponto flutuante armazenado como char
NUMERIC[(M,D)]			O mesmo que DECIMAL
DEC[(M,D)]			O mesmo que DECIMAL

OBSERVAÇÕES:

- As opções entre colchetes ([e]) são opcionais;
- Dentre os tipos que se ajustam aos dados a serem inseridos, escolha sempre o de menor tamanho;
- Para dados do tipo inteiro você pode usar a opção **UNSIGNED** para especificar inteiros positivos ou zero;

- *M especifica o tamanho máximo de exibição;*
- *D especifica o número de casas decimais. O valor máximo de D é 30 ou M-2;*
- *Tanto para números inteiros como para números de ponto flutuante você pode especificar a opção **ZEROFILL** que preenche os números com zeros iniciais. Colunas especificadas com ZEROFILL são automaticamente configuradas como UNSIGNED;*

Tipos de dados Data/Hora

TIPO	INTERVALO	DESCRIÇÃO
DATE	1000-01-01 a 9999-12-31	Data. Exibido como YYYY-MM-DD
TIME	-838:59:59 a 838:59:59	Hora. Exibido como HH:MM:SS
DATETIME	1000-01-01 00:00:00 a 9999-12-31 23:59:59	Data e hora. Exibido como YYYY-MM-DD HH:MM:SS
TIMESTAMP[M]	1970-01-01 00:00:00 a algum momento em 2037. Depende do limite do sistema operacional	Registro de data e hora útil para transações. Os formatos de exibição podem ser: TIMESTAMP YYYYMMDDHHMMSS TIMESTAMP (14) YYYYMMDDHHMMSS TIMESTAMP (12) YYMMDDHHMMSS TIMESTAMP (10) YYMMDDHHMM TIMESTAMP (8) YYYYMMDD TIMESTAMP (6) YYMMDD TIMESTAMP (4) YYMM TIMESTAMP (2) YY
YEAR[2]	70 a 69 (1970 a 2069)	Ano
YEAR[4]	1901 a 2155	Ano

Tipos de dados String

TIPO	INTERVALO	DESCRIÇÃO
[NATIONAL] CHAR(M) [BINARY]	0 a 255 caracteres	String de comprimento fixo M . NATIONAL especifica que o conjunto de caracteres padrão (ANSI SQL) será utilizado. BINARY especifica que os dados devem ser tratados de modo a não haver distinção entre maiúsculas e minúsculas (o padrão é distinguir).
CHAR	1	O mesmo que CHAR(1)
[NATIONAL]	1 a 255	String de comprimento variável
VARCHAR(M) [BINARY]	Variável	String de tamanho variável. O mesmo que [BINARY].
TINYBLOB	0 a $2^8 - 1$ (255)	BLOB pequeno
TINYTEXT	0 a $2^8 - 1$ (255)	TEXT pequeno
BLOB	0 a $2^{16} - 1$ (65535)	BLOB normal
TEXT	0 a $2^{16} - 1$ (65535)	TEXT normal
MEDIUMBLOB	0 a $2^{24} - 1$ (16777215)	BLOB médio

MEDIUMTEXT	0 a $2^{24} - 1$ (16777215)	TEXT médio
LONGBLOB	0 a $2^{32} - 1$ (4294967295)	BLOB longo
LONGTEXT	0 a $2^{32} - 1$ (4294967295)	TEXT longo
ENUM('valor1','valor2',...)	0 a 65535	Armazenam um dos valores listados ou NULL
SET('valor1','valor2',...)	0 a 64	Armazenam um ou mais dos valores listados ou NULL

OBSERVAÇÕES:

- *CHAR e VARCHAR armazenam Strings de comprimento fixo e variável respectivamente. VARCHAR trabalha mais lento.*
- *TEXT e BLOB armazenam textos grandes ou objetos binários (figuras, som, etc.). TEXT diferencia maiúsculas de minúsculas.*

Conceitos básicos sobre banco de dados relacional

Tabelas relacional

Um banco de dados relacional é formado por tabelas.

numeroDoCliente	Nome	Endereço	CEP
1	José Pedro	Rua 9, Nr 125, Centro	13153-689
2	Cristina da Silva	Rua Dr. Almeida, Nr 457, Campinas	13384-536
3	Fabio Júlio Ap.	Rua Arlindo Costa, Nr 15, Nova Veneza	87032-000
4	Dennis Oliveira	Rua Violeta, 156, Cidade Velha	15687-320

Cada **coluna** da tabela armazena um tipo de dado e representa um **campo** do banco de dados. Cada **linha** armazena os dados de um cliente e representa um **registro**.

Chave primária

Cada tabela deve ter um campo que identifica o registro. Os valor depositado neste campo para cada registro deve ser único, ou seja, não devem haver dois ou mais registros que tenham os mesmos dados armazenados. Este campo é chamado de chave primária. Na tabela de exemplo mostrada acima, a chave primária é o campo **numeroDoCliente**. Uma chave pode ser composta de mais de um campo na tabela.

Chave estrangeira

Uma tabela também pode fazer relacionamento com a chave de outra tabela, quando isto acontece a chave da outra tabela é chamada **chave estrangeira**.

Exemplo:

Tabela de Pedidos:

numeroDoPedido	numeroDoCliente	Valor	Data
1	5	200,00	050603
2	3	50,00	050603
3	1	175,00	060603
4	2	300,00	060603

Esta tabela faz relacionamento com a tabela de clientes pelo campo **numeroDoCliente**. A chave primaria desta tabela é o campo **numeroDoPedido** e a chave estrangeira é o campo **numeroDoCliente**. A função da Chave estrangeira nesta tabela, é informar quantos clientes compraram produtos na tabela de pedidos.

Relacionamentos

As chaves estrangeiras representam um relacionamento entre as tabelas. Existem três tipos de relacionamentos:

- de um para um;
- de um para muitos; e
- de muitos para muitos

Um para Um: significa que um registro em uma tabela só se relaciona com um registro na outra tabela.

Um para Muitos: significa que um registro em uma tabela relaciona-se a muitos registros na outra tabela.

Muitos para Muitos: significa que muitos registros de uma tabela relacionam-se a muitos pedidos da outra tabela.

Aula 2 - Instalação do MySQL

Existem versões do banco de dados MySQL, atualmente desenvolvidas para todos os sistemas operacionais conhecidos, como, Windows, Linux, Mac OS, entre outros.

Neste livro vamos demonstrar como instalar e utilizar o banco de dados MySQL no sistema operacional Windows, mas nada impede que o mesmo, possa ser instalada em outros sistemas para aprendizagem.

O download do programa do servidor de banco de dados MySQL pode ser feito diretamente do site oficial MySQL.com e ser instalado sem dificuldades no Windows, podendo escolher a versão para sistema operacional 32-bit ou 64 bit, dependendo do seu sistema. Mas em vez disto, vamos instalar o servidor MySQL através de um kit de programas de desenvolvimento Web onde, um dos programas é o MySQL. Escolhemos utilizar este pacote de programas, por já vir com o servidor apache para rodar páginas dinâmicas PHP e também o aplicativo Web, phpMyAdmin, aplicação esta, que vamos utilizar neste livro.

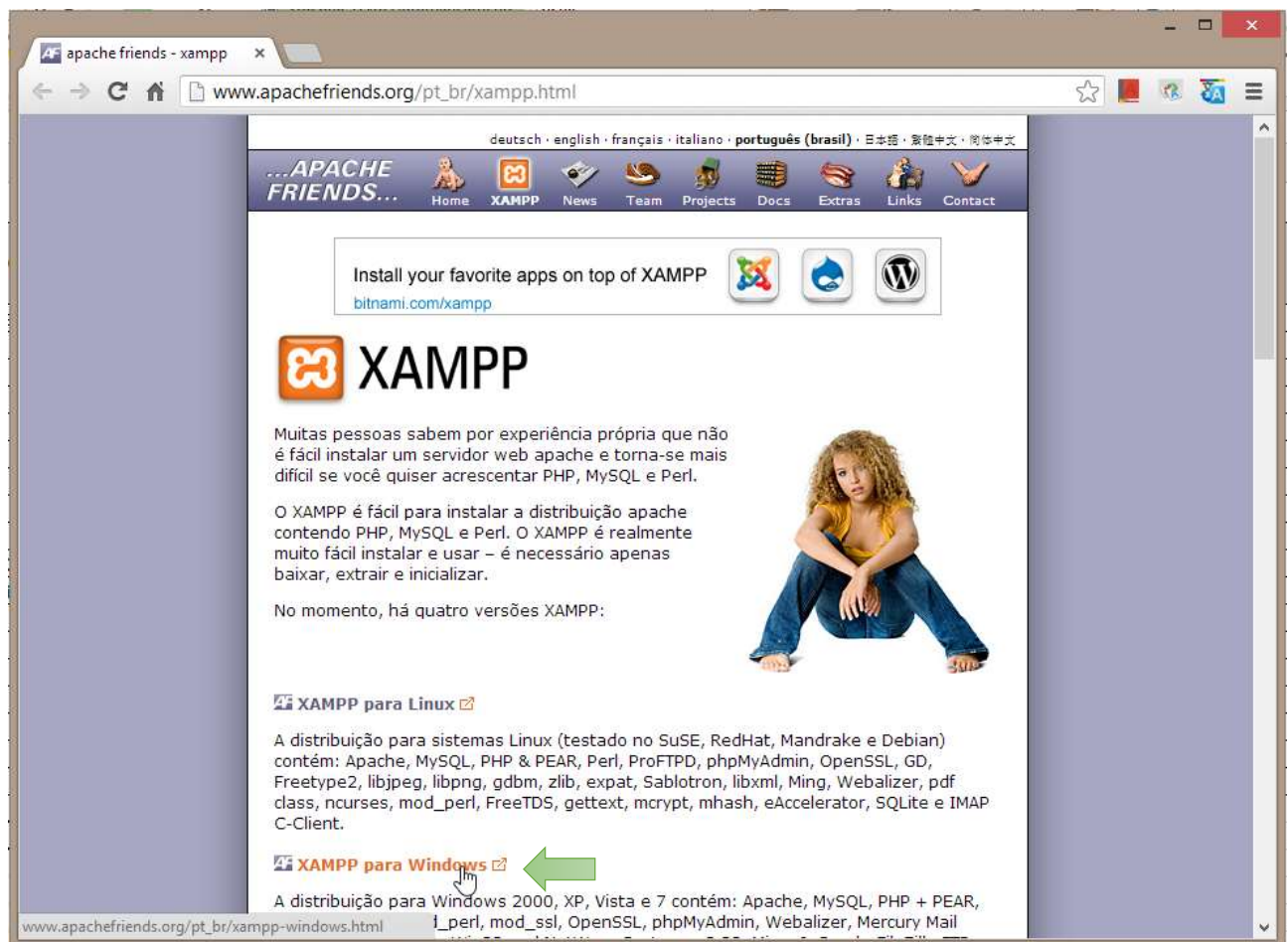
O que é o XAMPP?

O XAMPP é um pacote de distribuição de programas de desenvolvimento Web. Ao baixar e instalar o XAMPP, o mesmo instala e configura automaticamente o apache, servidor de páginas web dinâmicas como, o PHP e o mais importante para nós, o servidor de banco de dados MySQL atualmente na versão 5.1 no XAMPP.

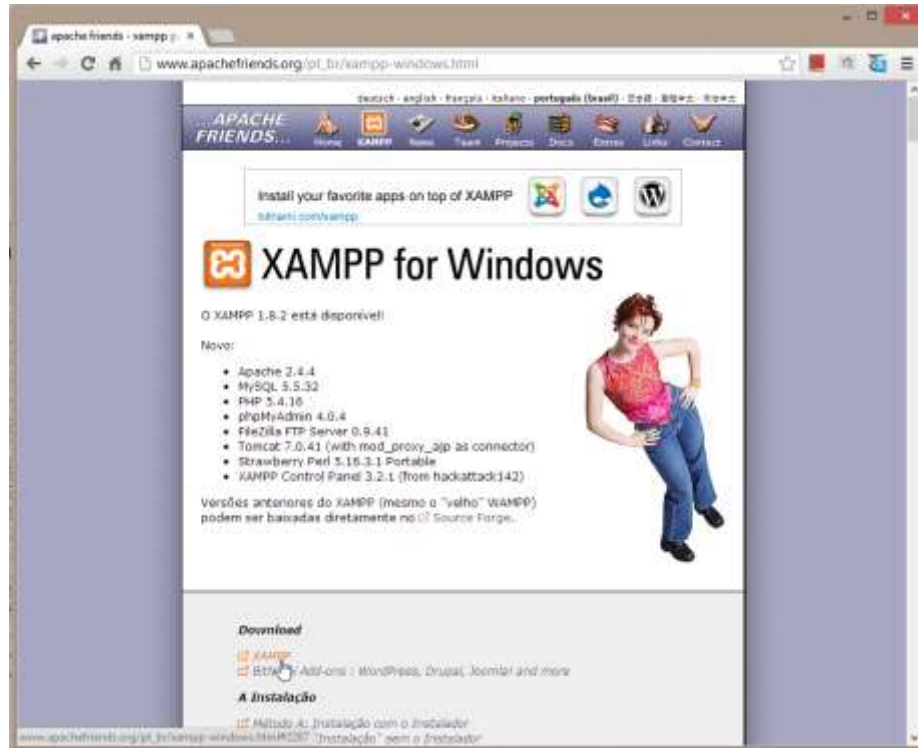
Instalando o pacote de programas XAMPP

O Download do pacote de programas XAMPP pode ser feito através da página oficial do XAMPP www.apachefriends.org/pt_br/xampp.html.

Acessando o link deste site, é possível baixar o XAMPP para o sistema operacional disponível que desejar. Clique no sistema operacional Windows.



Na página seguinte, abaixo do subtítulo Download, clique em XAMPP.



The screenshot shows a web browser window displaying the XAMPP for Windows website. The browser's address bar shows the URL www.apachefriends.org/pt_br/xampp-windows.html. The website features a navigation bar with the Apache Friends logo and various icons for Home, XAMPP, News, Tools, Projects, Docs, Events, Links, and Contact. Below the navigation bar, there is a banner that reads "Install your favorite apps on top of XAMPP" with icons for Joomla!, Drupal, and WordPress. The main heading is "XAMPP for Windows". A message states "O XAMPP 1.8.2 está disponível!". Under the heading "Novo:", a list of included software is provided:

- Apache 2.4.4
- MySQL 5.5.32
- PHP 5.4.16
- phpMyAdmin 4.0.4
- FileZilla FTP Server 0.9.41
- Tomcat 7.0.41 (with mod_proxy_ajp as connector)
- Strawberry Perl 5.15.3.1 Portable
- XAMPP Control Panel 3.2.1 (from hackattak142)

Below the list, it says "Versões anteriores do XAMPP (mesmo o 'velho' WAMP) podem ser baixadas diretamente no [Source Forge](#)." There is also a "Download" section with a link to the XAMPP installer and a note about additional software like WordPress, Joomla!, and Drupal. At the bottom, there is a section titled "A Instalação" with a link to the installation guide.

The screenshot shows a web browser window displaying the XAMPP download page for Windows. The page is titled "Download" and provides information about XAMPP versions and download options.

Download

XAMPP

Você pode baixar o XAMPP para Windows em três diferentes variações:

- Instalador**
Fácil e Seguro: XAMPP com um confortável instalador.
- Arquivo ZIP**
Para punstas: XAMPP em um arquivo ZIP.
- Arquivo 7zip**
Econômico: XAMPP em um pequeno arquivo 7ip.

XAMPP for Windows 1.8.2, 26.6.2013

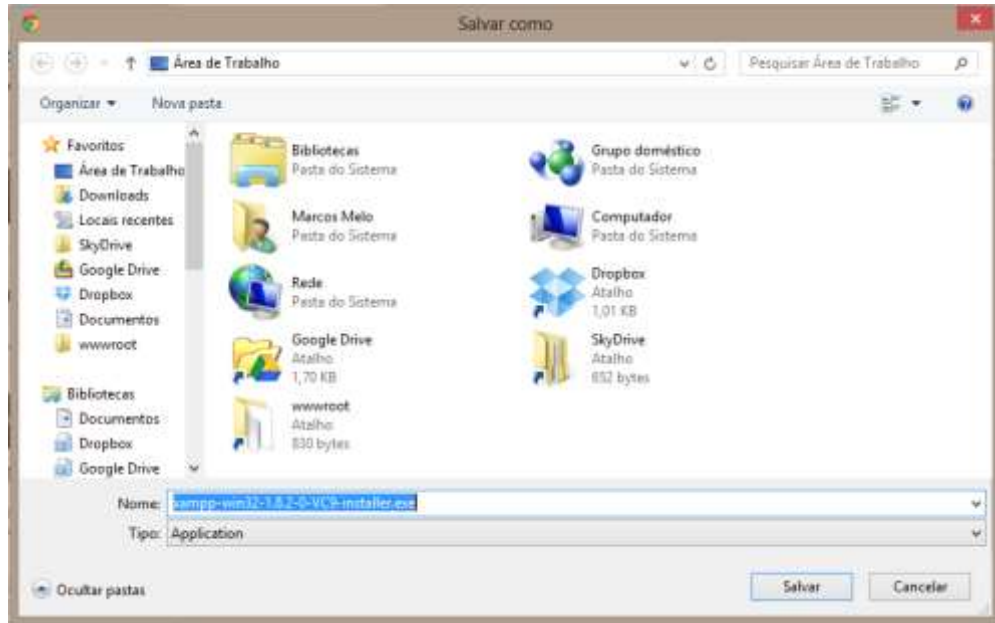
Versão	Tamanho	Conteúdo
XAMPP Windows 1.8.2		Apache 2.4.4, MySQL 5.5.32, PHP 5.4.16, phpMyAdmin 4.0.4, XAMPP Control Panel 3.2.1, Webalizer 2.23-04, Mercury Mail Transport System v4.62, FileZilla FTP Server 0.9.41, Tomcat 7.0.41 (with mod_proxy_ajp as connector), Strawberry Perl 5.16.3.1 Portable For Windows 2000, XP, Vista, 7, 8.
Instalador	102 MB	Instalador MD5 checksum: eee6115dfe06c9697259803cd86e521f
ZIP	180 MB	Arquivo ZIP MD5 checksum: 282da4b4d834779eeefc972454ec5d5
7zip	88 MB	Arquivo 7zip MD5 checksum: cf69db8307a2720de58c1ac92f251240

BITNami Add-ons : WordPress, Drupal, Joomla! and more

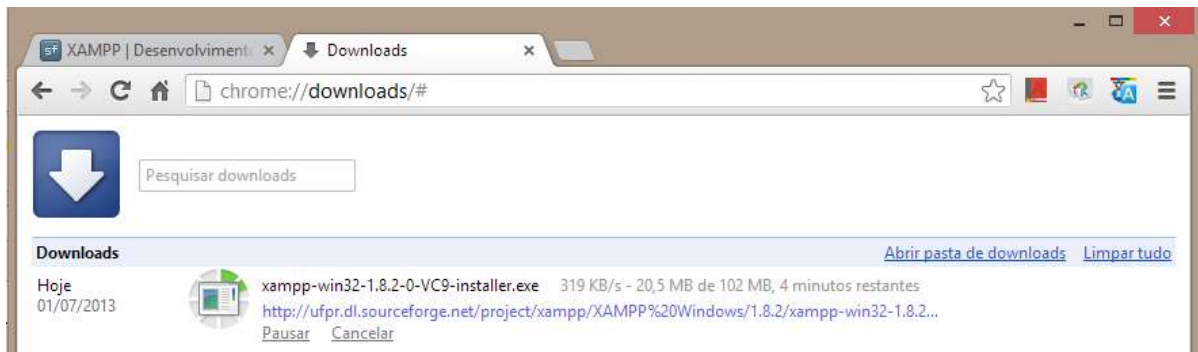
BitNami provides a free all-in-one tool to install Drupal, Joomla!, WordPress and many other popular open source apps on top of XAMPP. [Download BitNami XAMPP](#)

XAMPP portable

www.apachefriends.org/download.php5xampp-wm12-1.8.2... Tamanho Conteúdo



Aguarde o download do arquivo estar completo e em seguida execute o arquivo para instalá-lo.



A instalação é feita automaticamente sem a necessidade de maiores configurações.

Após a instalação do XAMPP o mesmo encontra-se na área de aplicações.

Dentro da pasta XAMPP, clique no ícone do programa “XAMPP Control”.

Dica: Crie um atalho se achar necessário arrastando o ícone do XAMPP Control para a área de trabalho.

Após executar o programa “XAMPP Control”, uma janela de controle aparecera para que você ative os programas do pacote. Clique nos botões “Start” dos três programas disponíveis, principalmente o do MySQL.

Após iniciar todos os serviços no XAMPP Control, o servidor local de banco de dados MySQL já estará funcionando.

É possível parar o serviço do servidor local de MySQL a qualquer momento pela janela de controle do XAMPP.

Acessando o banco de dados MySQL pela primeira vez

O sistema de banco de dados MySQL, não possui um ambiente visual gráfico nativo para acessá-lo e para a manipulação de seus dados. Basicamente podemos acessá-lo em modo texto via terminal de comando caso não tenhamos nenhum programa cliente de acesso a ele. Não há nenhum problema em acessar o banco de dados via terminal de comando, porém, por uma gestão de flexibilidade e agilidade, é essencial o uso de programas clientes de conexão ao servidor de banco de dados MySQL, que sejam de ambiente gráfico.

Ao instalar o XAMPP um desses programas clientes de acesso ao banco de dados MySQL é instalado automaticamente e é um dos mais usados pelos servidores de hospedagem de sites por ser em formato Web, ou seja, funciona direto no servidor Web e pode ser acessado em qualquer lugar do mundo pela internet em um browser. Este programa é o phpMyAdmin e vamos abordar sua utilização previa em seguida.

Acessando o phpMyAdmin

Para acessar o phpMyAdmin, lembre-se que, além do servidor MySQL o servidor Apache tem que estar ativo também no painel XAMPP Control. O phpMyAdmin foi desenvolvido em linguagem php, por isso da necessidade de ter o apache ativado.

Basicamente, para acessar o phpMyAdmin, basta digitar na barra de endereços do navegador (Browser) o link local <http://localhost/xampp/> e em seguida clicar no link do painel de opções do XAMPP em phpMyAdmin ou é possível acessa-lo diretamente pelo link <http://localhost/phpMyAdmin/>.



XAMPP for Mac OS X

English / Deutsch / Français / Nederlands / Polski / Italiano / Norsk / Español / 中文 / Português (Brasil) / 日本語

XAMPP

Bem-Vindo

Status

Segurança

Documentação

Componentes

Demos

Coleção de CD

Biorbimo

Livro de Visitas

Arte Instantânea

Arte Flash

phpinfo()

Agenda de Telefones

Ferramentas

phpMyAdmin

Webalizer

© 2002-2008
...APACHE
FRIENDS...

Bem vindo ao XAMPP para Mac OS X 1.7.3!

Congratulações:

Você instalou corretamente o XAMPP em seu sistema!

Você pode agora iniciar a utilização do Apache e outros aplicativos. Primeiramente tente verificar o »Status« no menu lateral para ter certeza que tudo está funcionando corretamente.

Antes de efetuar este teste, você poderá visualizar os exemplos abaixo do link de teste.

Caso deseje iniciar programando PHP ou Perl (ou qualquer outro r ;) por favor, dê uma olhada no [Manual do XAMPP](#) primeiro para obter melhores informações sobre a instalação de seu XAMPP.

Boa sorte,
Kristian Marcroft, Florian Pollini, Christian Speich & Team

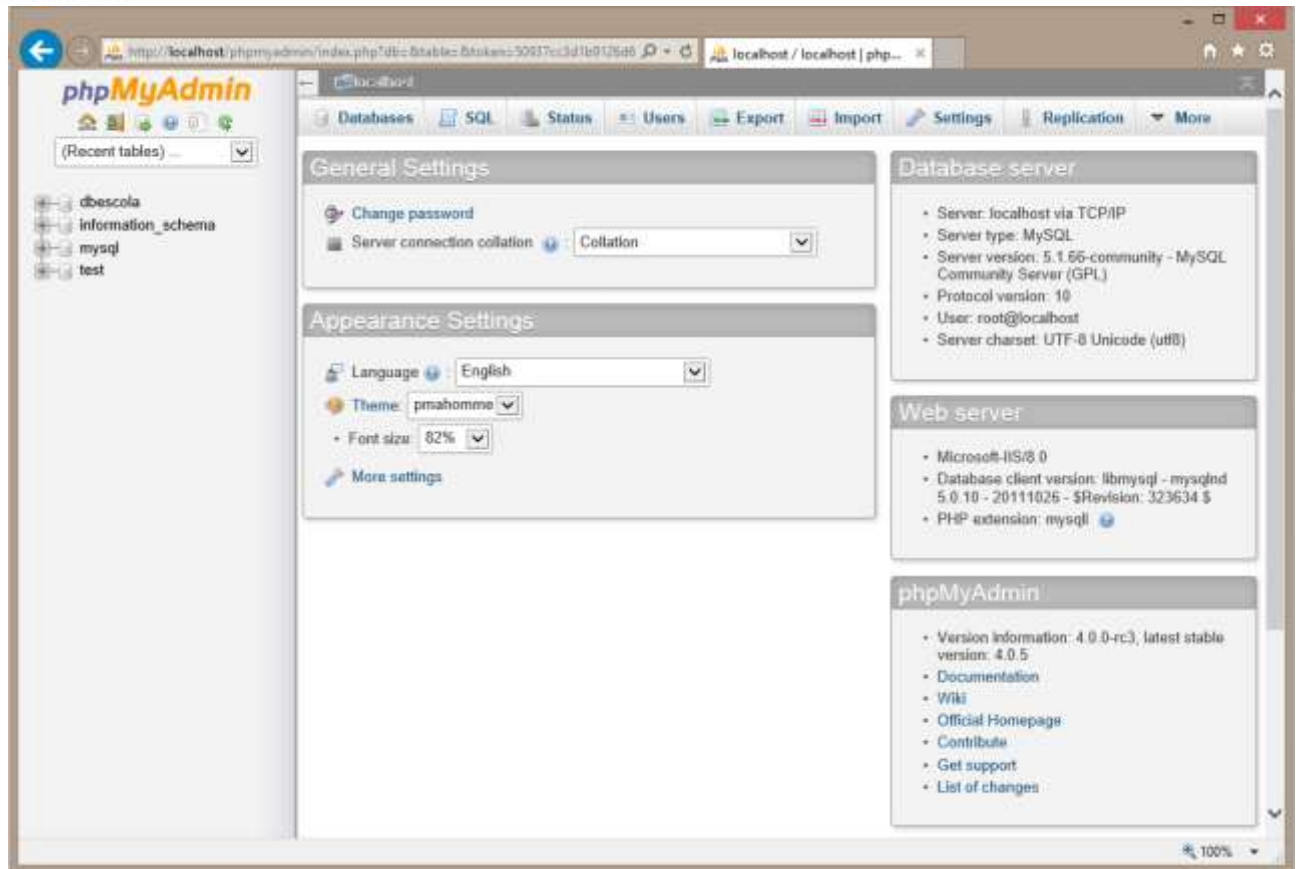


Menu de controle do XAMPP para acessar suas opções



Clique no link [phpMyAdmin](#) para acessá-lo

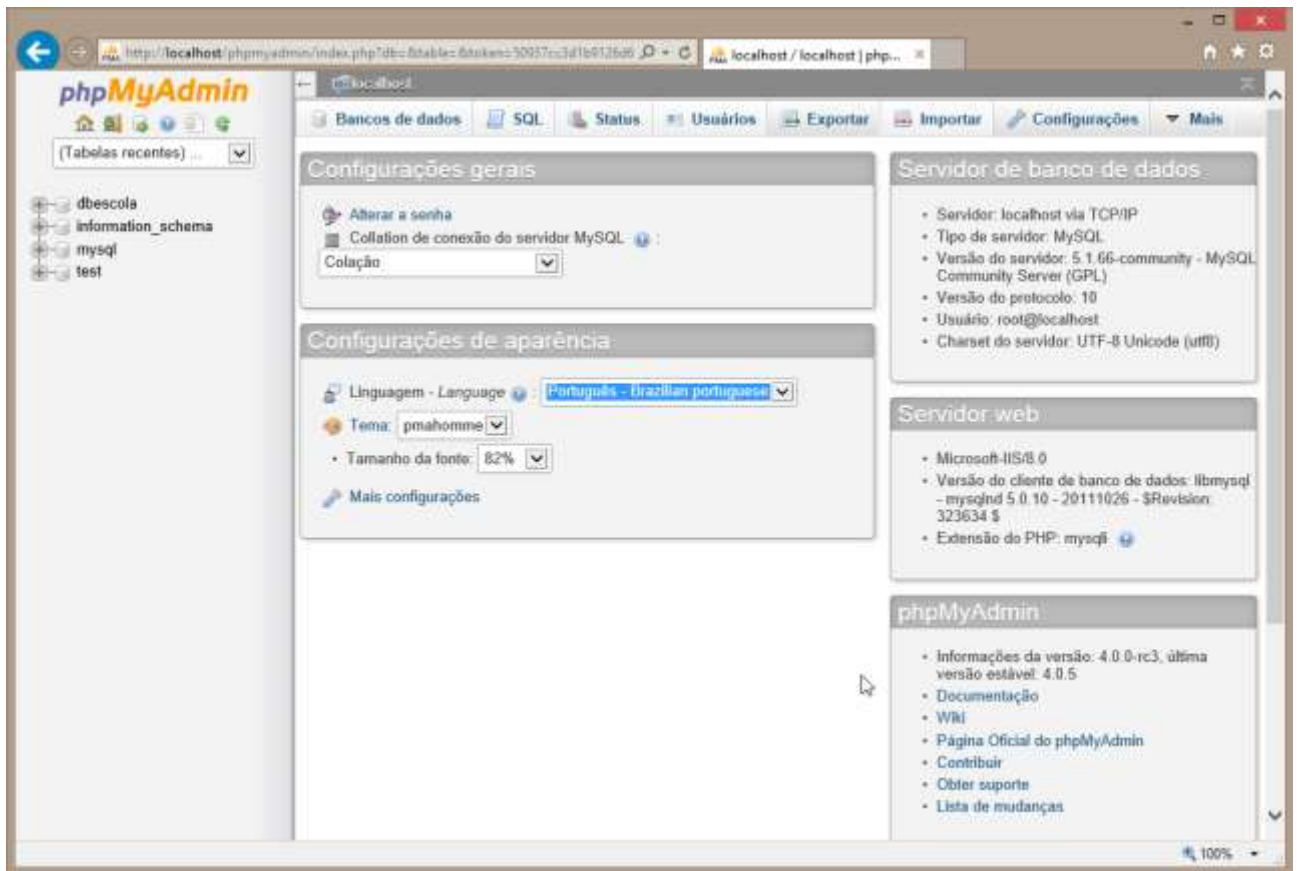
Observe a janela do link do ambiente gráfico do phpMyAdmin pelo seu Browser.



Janela do ambiente gráfico do phpMyAdmin

Podemos observar que, o ambiente gráfico encontra-se no idioma inglês, o que não é problema nenhum pra quem quer se aventurar no mundo de programação, já que tudo está relacionado a este idioma. Mas, para começar a usar pela primeira vez e não ter muita dificuldade com entendimento no idioma inglês, vamos mudar a linguagem do programa para português.


Nas opções de interface, no item “Language”, selecione a opção “Português – Brazilian Portuguese”.

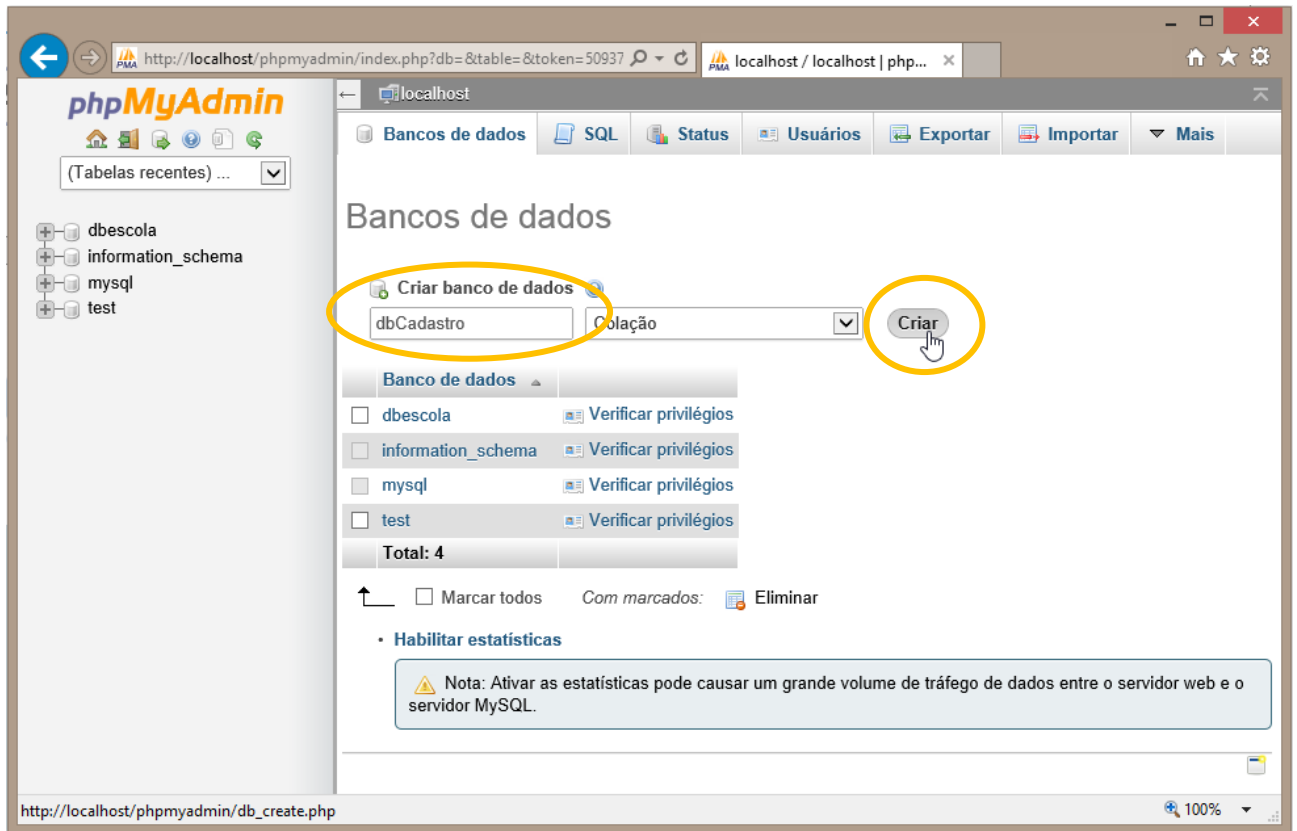


Mudando o idioma de inglês para a linguagem Português.

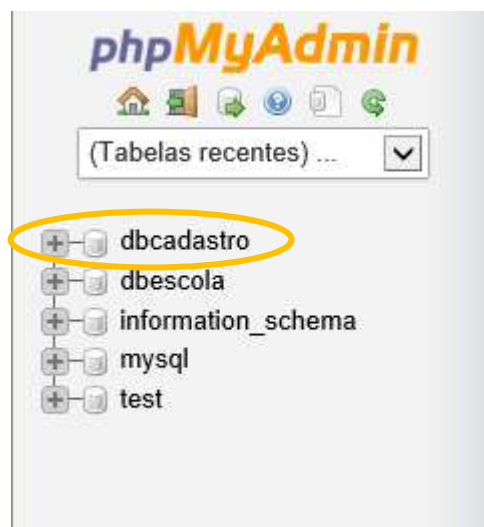
Criando seu primeiro banco de dados

Vamos criar um banco de dados pelo método gráfico do phpMyAdmin, ou seja, não digitaremos uma só linha de código da linguagem SQL. Este é só o primeiro de vários bancos de dados que você criará. A maioria dos próximos bancos de dados serão criados em linguagem de programação SQL. Bom, vamos lá;

1. Com o phpMyAdmin aberto, clique o ícone , para garantir que esteja na tela principal onde criamos novos bancos de dados. No menu de opções a direita selecione **Banco de dados**.
2. Vamos criar um banco de dados novo com o nome “dbCadastro” digitando na área **Criar base de dados**, como informado abaixo. Em seguida clique no botão **Criar**.



Criando um banco de dados.

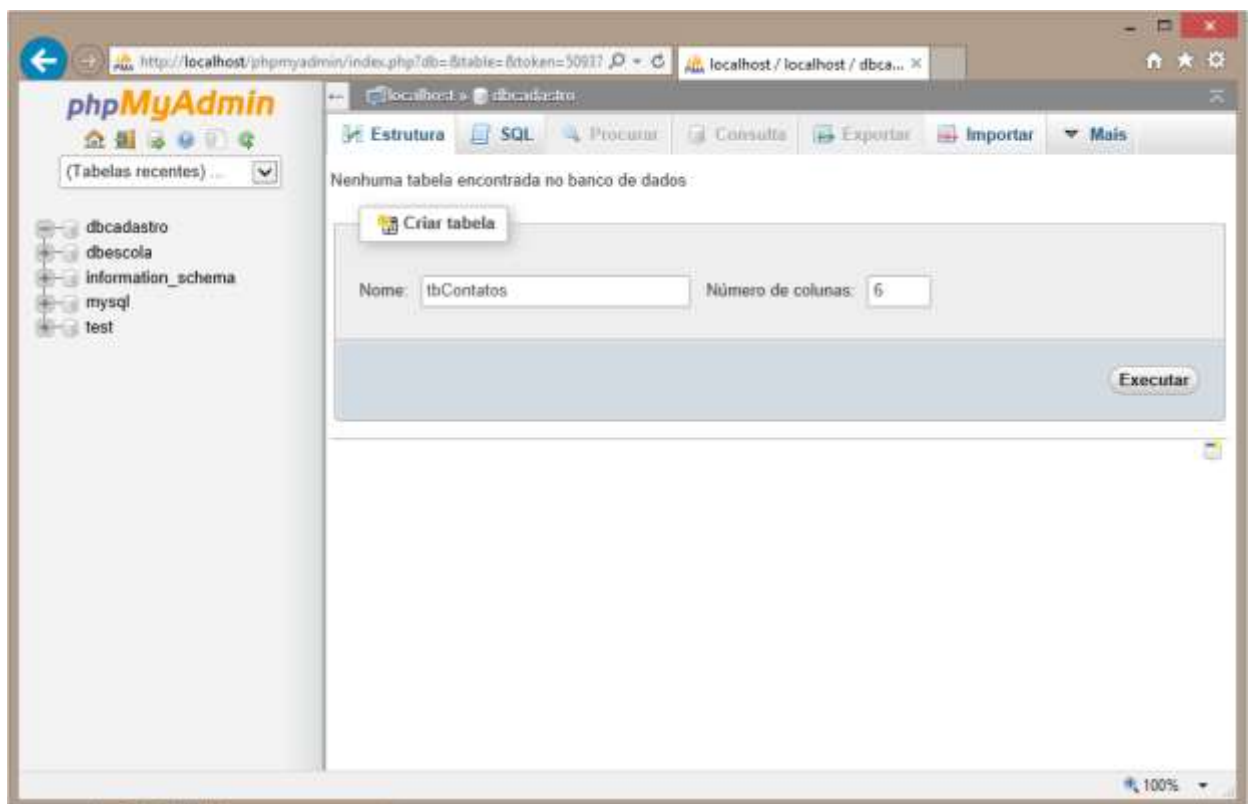


Notificação da criação do banco de dados criado com sucesso.

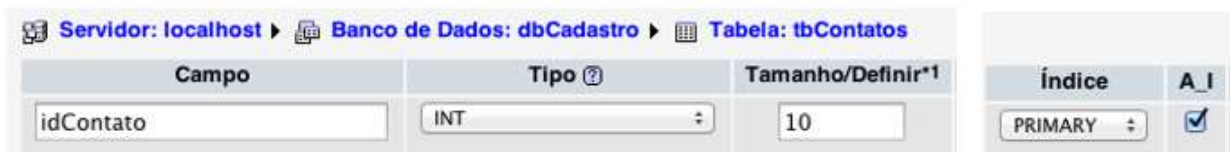
Criando a primeira tabela

Após ter criado o banco de dados, o mesmo se encontra na lista de banco de dados disponíveis do lado esquerdo. Clique nele para selecioná-lo para que possamos criar tabelas de dados que iram compô-lo. Vamos criar uma tabela genérica de contatos somente como exemplo. Veja a seguir os passos para criar esta tabela;

1. Como ainda não há nenhuma tabela criada, o phpMyAdmin já mostra a tela de criação de nova tabela. Para criar uma nova tabela no phpMyAdmin pelo ambiente gráfico, informe o nome “tbContatos” para a tabela e informe o número de campos 6 e em seguida clique em **Executar**.



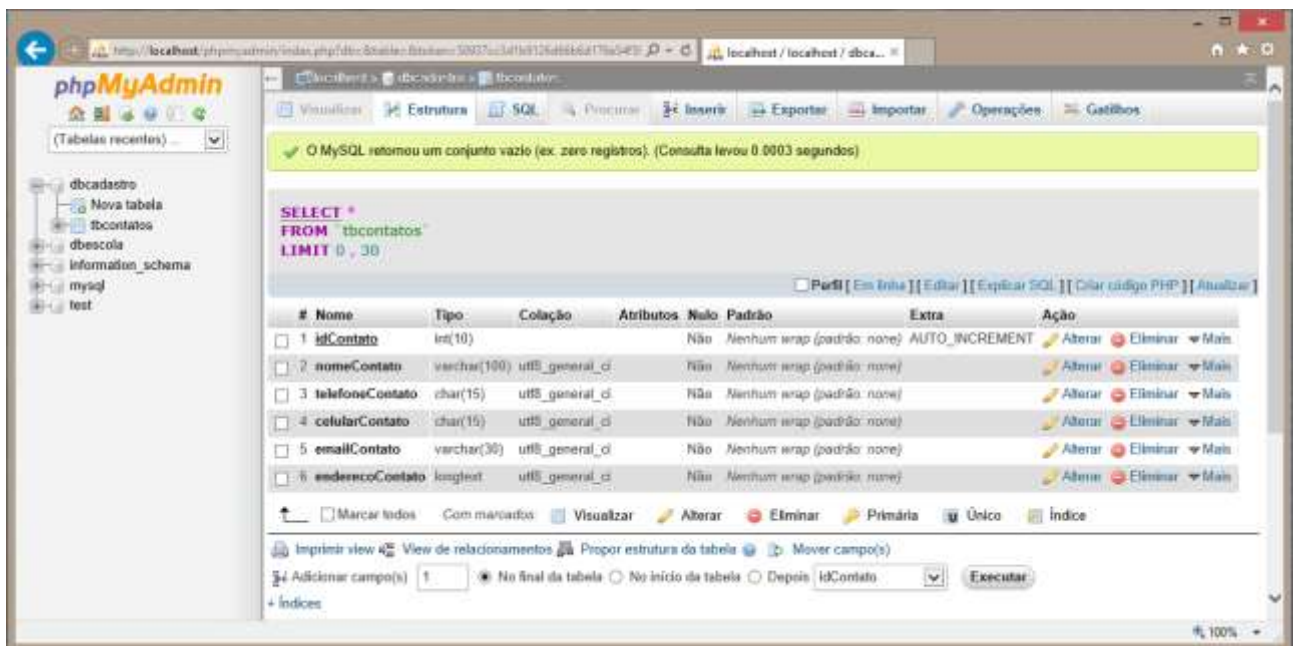
2. Agora vamos criar os campos da tabela. São 6 linhas com propriedades para cada campo. No primeiro campo definimos a chave primária. Digite conforme a imagem abaixo;



3. Crie os próximos campos conforme imagem abaixo;

Campo	Tipo ?	Tamanho/Definir*1
idContato	INT	10
nomeContato	VARCHAR	100
telefoneContato	CHAR	15
celularContato	CHAR	15
emailContato	VARCHAR	30
enderecoContato	LONGTEXT	

4. Após digitar todos os campos, clique em **Salvar** para criar a tabela de Alunos.



Notificação da tabela tbContatos criado com sucesso.

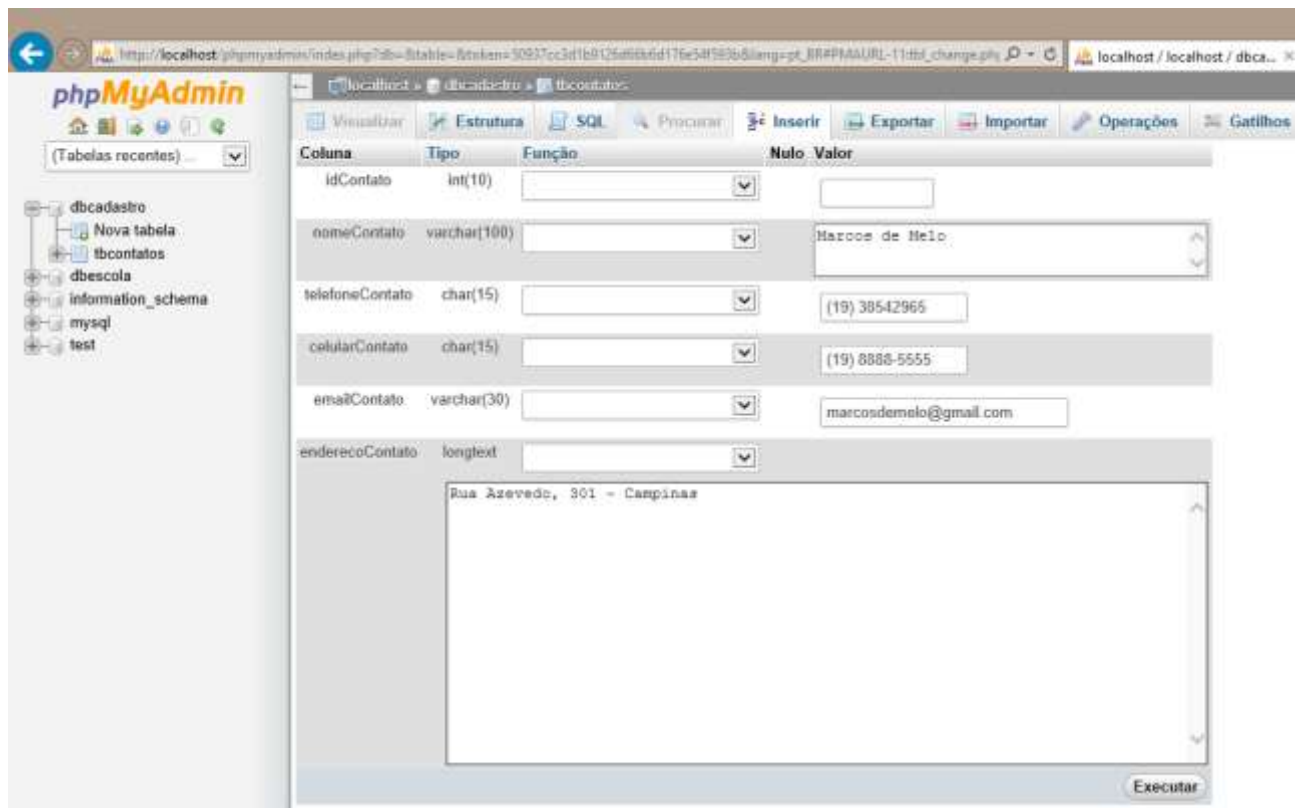
Inserindo dados na tabela

Agora que criamos a tabela tbContatos, vamos cadastrar alguns contatos pessoais nela.

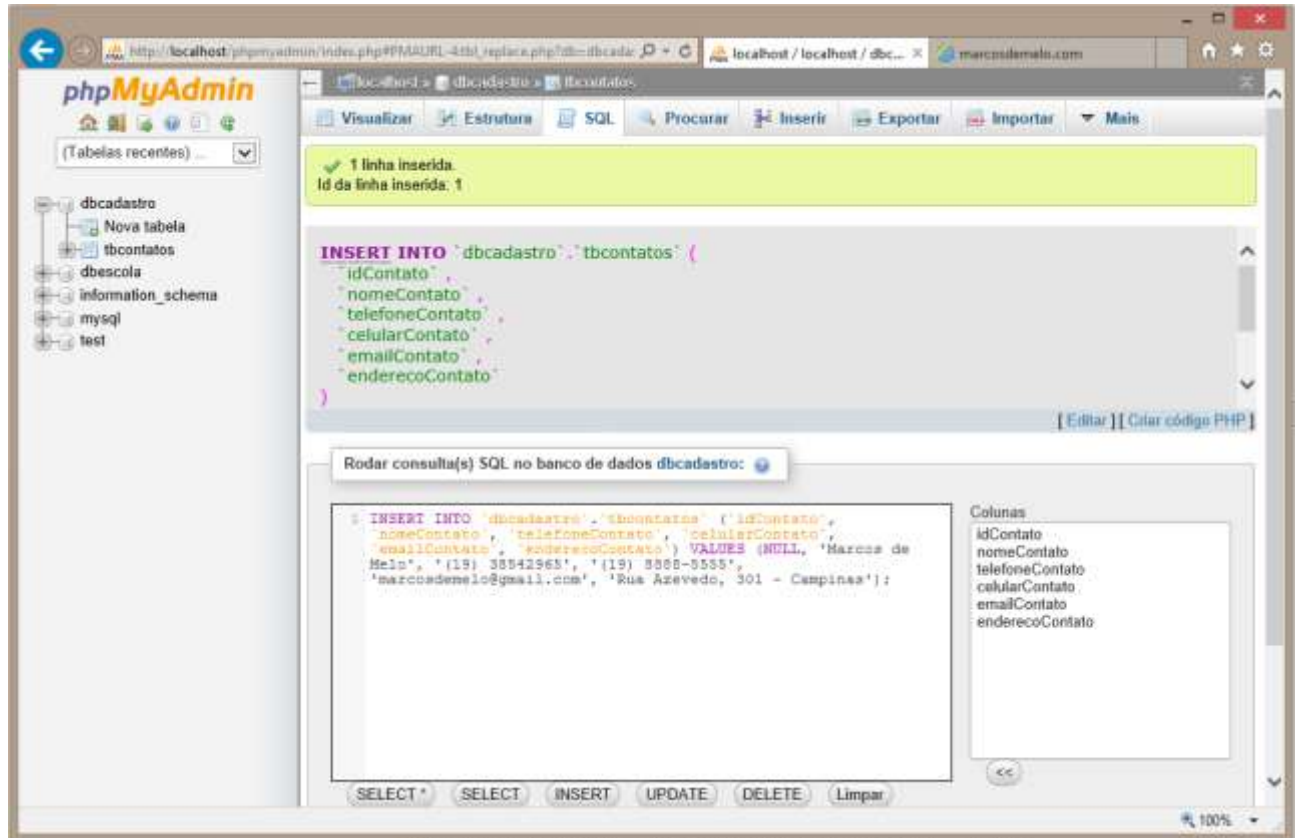
1. Clique na guia **Inserir** e comece a digitar os dados de novos Contatos, conforme exemplo mostrado abaixo.

Obs: Repare que, o campo idContato não deve ser preenchido, o mesmo será preenchido automaticamente pelo sistema, pois é auto incrementado.

2. É possível digitar as informações de 2 registros por vez. Clique em **Executar** para inserir os registros.



Cadastrando informações de registros.



Notificações de registros cadastrados com sucesso.

Visualizando os registros cadastrados nas tabelas

Para visualizar os dados após o cadastro, clique na guia “Visualiza”. É preciso estar com a tabela desejada selecionada.

The screenshot shows the phpMyAdmin interface. A green arrow points to the 'SQL' tab. The SQL query entered is: `SELECT * FROM tbcontatos LIMIT 0, 30`. The results table shows one record for 'Marcos de Melo'.

	idContato	nomeContato	telefoneContato	celularContato	emailContato	enderecoContato
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Remover	1	Marcos de Melo	(19) 38542965	(19) 8888-5555	marcosdemelo@gmail.com	Rua Azevedo, 301 - Campinas

Atividades

1 – Crie mais uma tabela no banco **dbCadastro** chamada “**tbFuncionarios**”. Siga os mesmos procedimentos executados acima para cria-la.

Campos para a tabela;

idFuncionario, nomeFuncionario, cpfFuncionario, emailFuncionario, telefoneFuncionario.

Aula 3 – Criação de Bancos e Tabelas

Nesta aula vamos começar a manipular um banco de dados efetivamente, em linha de comando, por tanto, os bancos de dados, tabelas e campos, que você criou no modo gráfico no programa phpMyAdmin, foi a última vez. Usaremos códigos e mais códigos SQL para criá-los daqui para frente. Só utilizaremos programas gráficos para facilitar a criação de múltiplos comandos.

O Workbench (GUI tool)

Já falamos um pouco sobre o phpMyAdmin, programa via Web, desenvolvido em linguagem PHP para acesso e manipulação de dados MySQL.

Mas nesta aula faremos uma abordagem ao programa WorkBench (GUI Tool). Criado pelos próprios desenvolvedores do MySQL.

O Workbench é um software de uso gratuito para conectar local ou remotamente, um servidor de bancos de dados MySQL, com recursos bastante eficientes de gerenciamento e manipulação de dados.

Instalando e usando o Workbench (GUI tool)

Baixe e instale o programa Workbench direto do site oficial www.mysql.com na versão para Windows.

Atenção!

Caso os computadores na sala de aula, já estejam com o programa Workbench instalado, ignore a instalação, mas reveja este tópico para futuras instalações como em seu computador pessoal e poder praticar em casa.

Por favor relate quaisquer erros ou inconsistências que você observa ao nosso banco de dados erros . Obrigada pelo vosso apoio!

MySQL Workbench 5.2.47

Selecione Plataforma:

Microsoft Windows Selecionar

Download	Version	Size	Action
Do Windows (x86, 32-bit), MSI Installer (mysql-workbench-gpl-5.2.47-win32.msi)	5.2.47	26.8M	Baixar
Do Windows (x86, 32-bit), ZIP Archive (mysql-workbench-gpl-5.2.47-win32-ripinstall.zip)	5.2.47	29.5m	Baixar

Nós sugerimos que você use o MD5 checksum e assinaturas GnuPG para verificar a integridade dos pacotes que você baixar.

Após clicar no botão Download, uma próxima página solicitará um cadastro para a ativação de uma conta no site da Oracle. Você pode ignorar o cadastro caso não queira clicando no link **“No thanks, just start my download”**. Logo em seguida o Download do instalador do programa começará.

Comece seu Download - mysql-workbench-gpl-5.2.47-win32.msi

Entrar agora ou Cadastre-se para criar uma conta gratuita.

Conta Oracle Web oferece as seguintes vantagens:

- Acesso rápido a MySQL de download de software
- Baixar White Papers e apresentações técnicas
- Postar mensagens nos Fóruns de Discussão do MySQL
- Informar e rastrear bugs no sistema bug MySQL
- Comentários na Documentação do MySQL

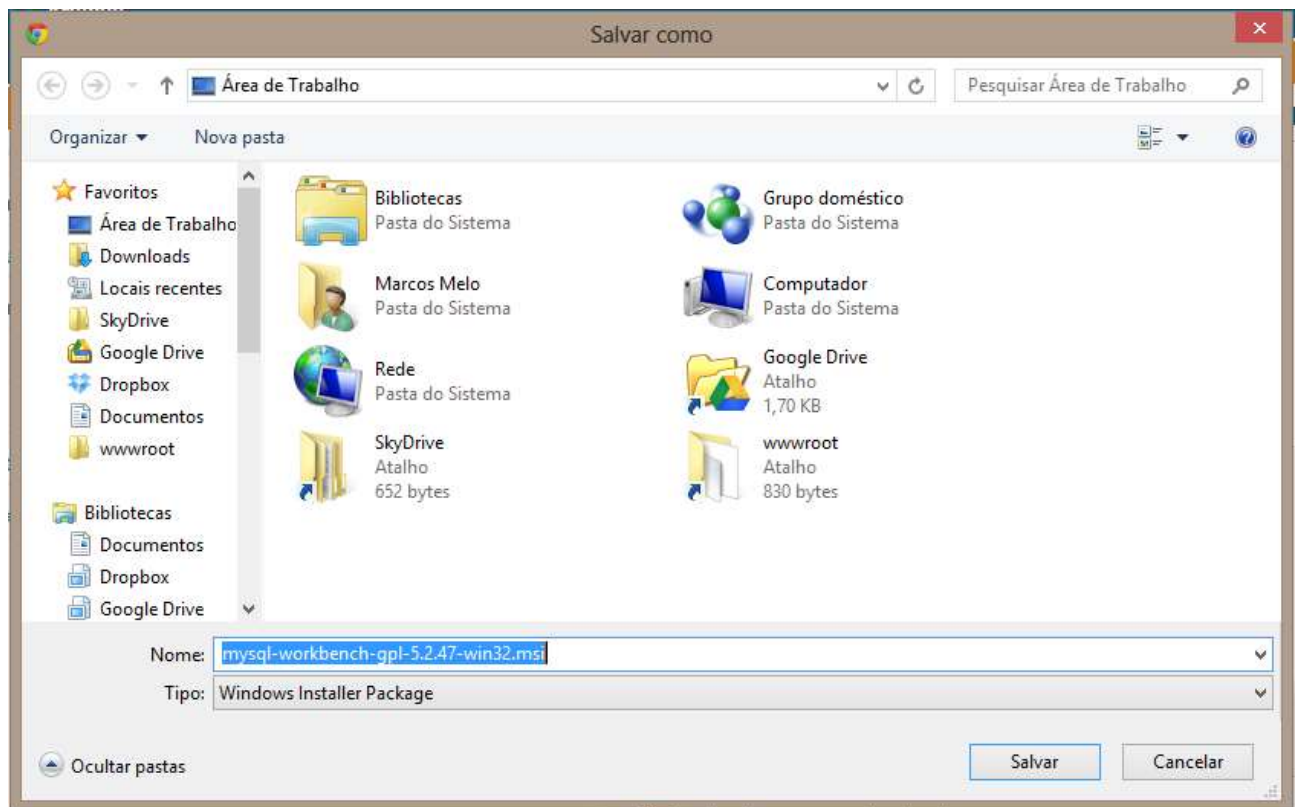
Entrar quando já tem uma conta do Oracle Web

Inscreva-se para criar uma conta do Oracle Web

MySQL.com está usando o Oracle SSO para autenticação. Se você já tiver uma conta do Oracle Web, clique no link Login. Caso contrário, você pode inscrever-se para criar uma conta gratuita clicando no sinal de # e seguindo as instruções.

Não, obrigado, basta começar meu download

Escolha a pasta desejada para salvar o Download e clique em **Save (Salvar)**.



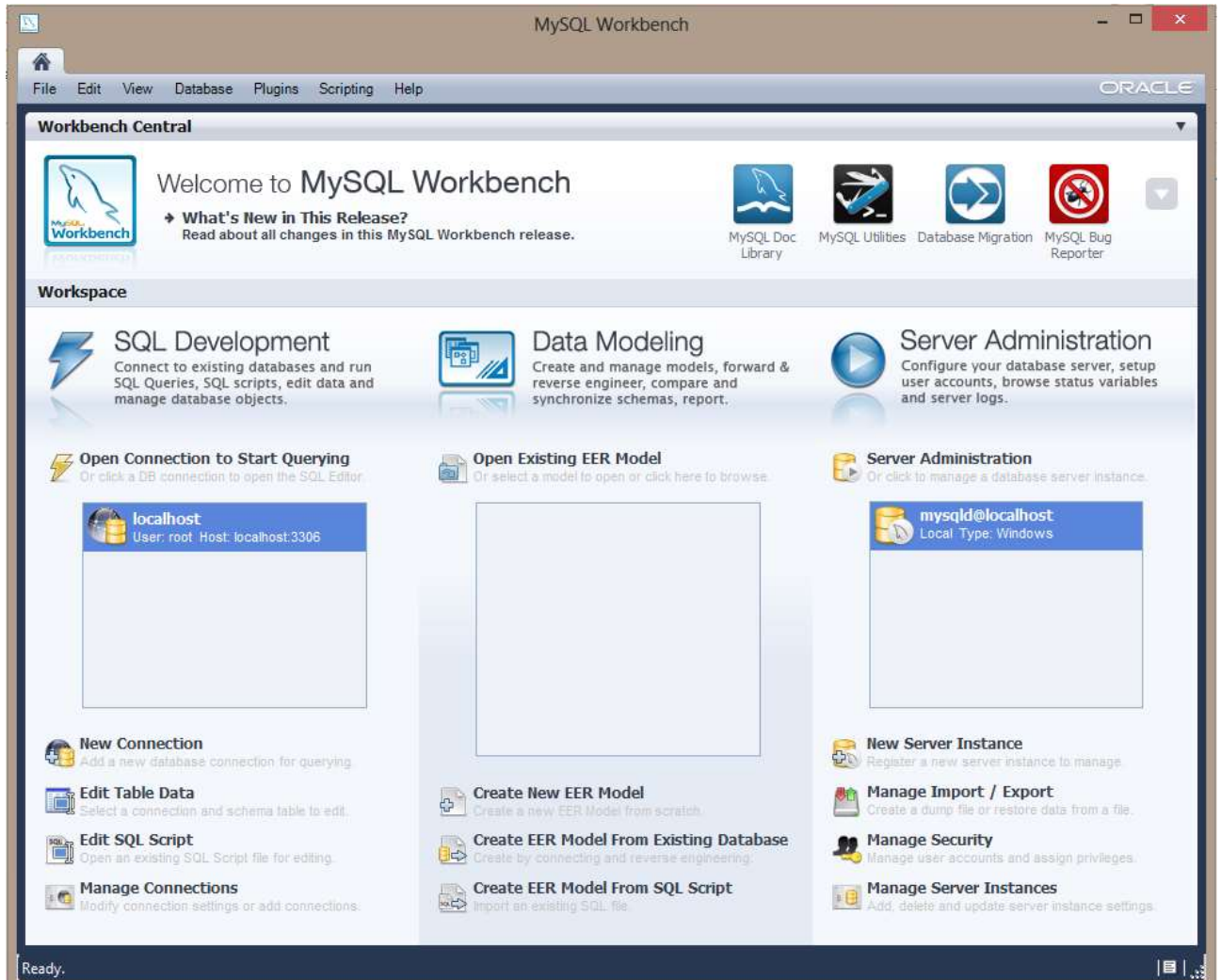
Após baixar o arquivo de instalação, execute-o clicando duas vezes neles.

Executando o Programa Workbench

Após instalar o Workbench em seu PC, o mesmo estará disponível.

Interface do programa

O Workbench reúne várias funções de gerenciamento e manipulação de banco de dados no MySQL. Estas funcionalidades são divididas em três partes; SQL Development, Data Modeling e Server Administration.



SQL Development

Nesta sessão podemos criar conexões a vários servidores MySQL, local ou remotos e a partir desta conexão, manipular os bancos de dados, como, criar e excluir bancos de dados, criar, editar e excluir tabelas, assim como manipulação dos dados ex: Seleção, Exclusão, atualização e inserção.

Data Modeling

Cria modelos de bancos de dados que explique as características de funcionamento e comportamento de um software a partir do qual ele será criado.

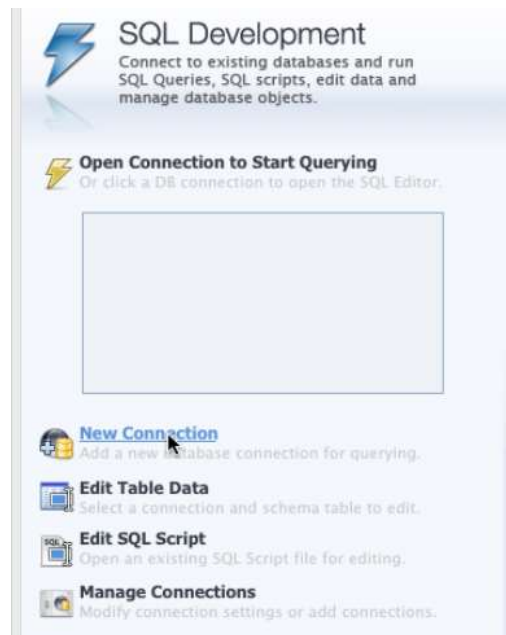
Server Administration

Conecta a um servidor MySQL especificado local ou remoto e aplicar ao mesmo, funções de administração e gerenciamento do servidor como, contas de usuários, importação e exportação de dados.

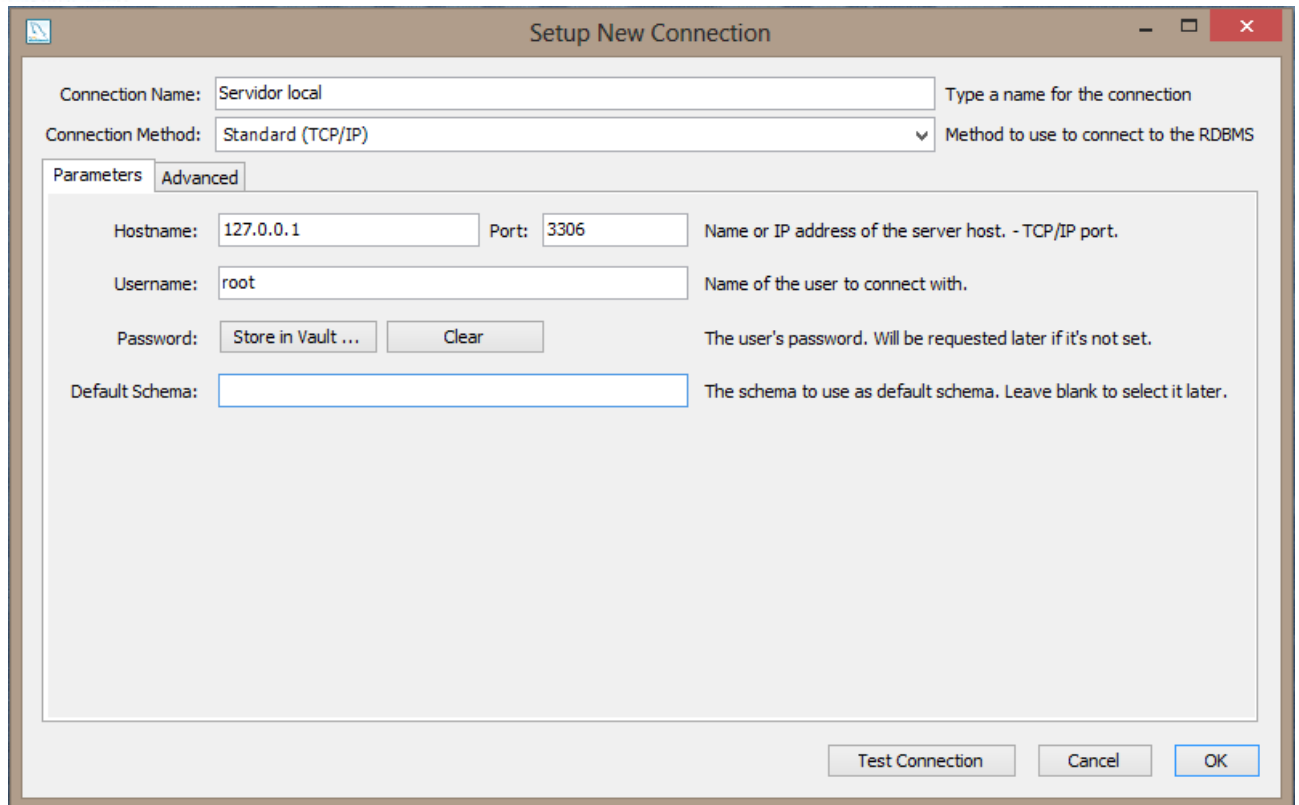
Criando uma nova instância de conexão

Vamos criar uma nova conexão ao nosso servidor local de banco de dados MySQL e a partir desta conexão manipular todos os bancos de dados que vamos criar daqui pra frente.

1. No lado esquerdo na **sessão SQL Development**, clique no link **New Connection** para adicionar uma nova conexão.



2. Na janela **Setup New Connection**, no campo **“Connection Name”** especificamos o nome para a conexão. Digite **“Servidor Local”**.
3. **Hostname** é o endereço IP ou DNS do servidor. Digite **“localhost”**.
4. **Username** usaremos o root usuário padrão com todos os privilégios do mysql. A senha (Password) do root está em branco.
5. Clique em **“Test Connection”** para testar se a conexão está funcionando. Em seguida clique OK para criar a conexão.



6. Após criar a conexão, a mesma encontra-se na lista de conexões. Clique duplamente nesta conexão que acabou de criar para acessar este servidor.



Criando banco de dados e suas tabelas

Agora vamos começar a criar, editar e manipular nossos bancos de dados efetivamente. Apesar de o Workbench também possuir um ambiente gráfico facilitado para a criação de banco de dados, não vamos criar nem editar nossos bancos de dados por este método, pois precisamos praticar como criar e manipular o banco de dados utilizando a linguagem SQL constantemente.

Observações

As informações especificadas entre "< ... >" informados nas sintaxes dos comandos, indicam os nomes dos objetos a serem criados como, bancos de dados e tabelas e são de preenchimento obrigatório.

Informações entre "[..]" são opcionais.

Criando um novo banco de dados

Para criar uma nova base de dados ou uma nova tabela de uma base de dados, usamos o comando sql `CREATE`. Veja a sintaxe para criar uma nova base de dados;

```
CREATE DATABASE <nome_do_banco>.
```

Vamos criar um banco de dados para um sistema escolar como exemplo e em seguida criaremos as tabelas dele.

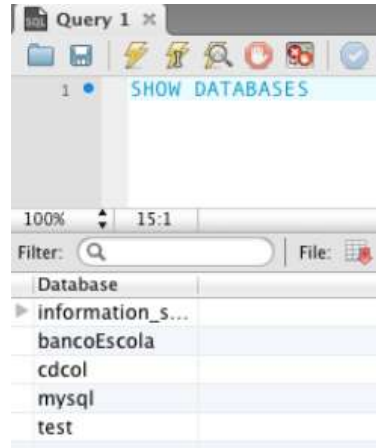
Digite o comando abaixo na área de comando SQL para criar o banco chamado "bancoEscola".

```
CREATE DATABASE bancoEscola
```

Visualizando bancos de dados

Para visualizar os bancos de dados existentes e verificar se o banco de dados "bancoEscola" foi realmente criado. Digite o comando abaixo.

```
SHOW DATABASES
```



Ativando um banco de dados

Para criar novas tabelas para o banco de dados “bancoEscola” sem precisar fazer referência a ele, devemos ativá-lo para uso. Use o comando abaixo para ativá-lo.

```
USE bancoEscola
```

Deletando um banco de dados

Para deletar um banco de dados, usamos o comando `DROP` no banco que se deseja excluir. Veja a sintaxe para excluir um banco de dados;

```
DROP DATABASE <nome_do_banco>
```

Crie um banco de dados exemplo com o nome “bancoTeste” e delete-o em seguida usando o comando abaixo;

```
DROP DATABASE bancoTeste
```

Criando tabelas

Vamos agora criar as tabelas para o banco de dados “bancoEscola”. A sintaxe para a criação de uma tabela é;

```
CREATE TABLE <nome_da_tabela> (<campo1> <tipo_de_dado>,<campo2> <tipo_de_dado>,...)
```

Vamos criar uma tabela de cadastro de alunos chamada `tbAlunos`. Digite o código abaixo para criá-la. Não se preocupe se não estiver entendendo nada do código abaixo, pois ele será explicado em detalhes em seguida.

```
CREATE TABLE tbAlunos(  
idAluno INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
nomeAluno VARCHAR(64) NOT NULL,  
endereçoAluno VARCHAR(30) NOT NULL,  
bairroAluno VARCHAR(20) NOT NULL,  
foneAluno VARCHAR(15) NOT NULL,  
emailAluno VARCHAR(20) NOT NULL,  
idadeAluno INT(3) NOT NULL  
)
```

Após ter digitado corretamente o código acima e executado o comando, a tabela deve ter sido criada normalmente. Mas vamos entender este código.

Primeiramente após a palavra reservada CREATE define-se o nome da tabela. O nome da tabela não deve conter espaços em branco e o limite de caracteres permitidos para o nome é 64.

Após o nome da tabela definimos as colunas de campos nos parâmetros da tabela entre parênteses "()". Os campos (colunas) são separados por vírgula ",", ". Veja a sintaxe da criação de uma coluna;

```
<nome_campo_coluna> <tipo_de_dado> [NULL | NOT NULL] [DEFAULT <valor>]  
[AUTO_INCREMENT] [PRIMARY KEY | INDEX]
```

O nome da coluna segue a mesma regra do nome da tabela sem espaços em nomes compostos e no limite de até 64 caracteres e depois define-se o tipo de dados que poderá ser aceito nesta coluna e os modificadores da coluna.

Mais informações sobre tipos de dados consulte a primeira aula no tópico "Tipos de dados".

Modificadores para as colunas

As colunas podem receber os seguintes modificadores:

- Not Null (NN);
- Default - valor por omissão - valor que a coluna recebe quando se insere null;
- Zerofill - usado nos valores numéricos - preencher com zeros à esquerda;
- Unsigned - usado nos valores numéricos - assume apenas valores positivos, o que duplica o maior valor possível;
- Auto_increment - usado para valores numéricos;

`NULL` e `NOT NULL` são valores opcionais, o padrão é `NULL`. Se mudar para `NOT NULL` o campo terá que ter preenchimento obrigatório.

A opção `AUTO_INCREMENT` aplicada no campo indicado, incrementa a entrada de dados automaticamente iniciando em 1. Somente um campo da tabela pode ter esta opção ativada, geralmente esta opção é aplicada ao campo chave primaria, ou seja, que possui a propriedade `PRIMARY KEY` ativada também onde o próprio sistema alimenta este campo, incrementando de um em um.

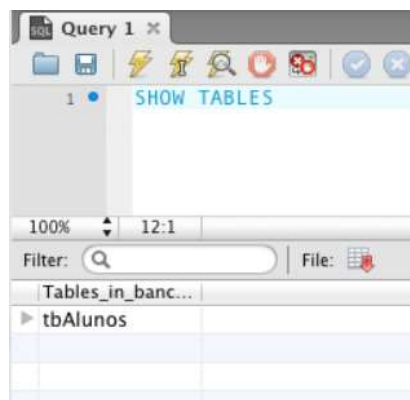
A propriedade `PRIMARY KEY` (chave primaria) é aplicada somente em um único campo na tabela. Tem como função, tornar o valor do campo, um identificador único de cada registro.

Exibindo as tabelas existentes no banco de dados

Para visualizar as tabelas existentes de uma banco de dados em uso, usamos o comando `SHOW TABLES`.

Digite o comando abaixo para visualizar a tabela `tbAlunos` criada recentemente no banco de dados `bancoEscola`.

```
SHOW TABLES;
```



Exibir descrições de uma tabela

Para visualizar em linha de comando os campos e suas descrições dentro de uma tabela podemos usar o comando `DESCRIBE`. Veja a sintaxe do comando;

```
DESCRIBE <nome_da_tabela>
```

Digite o comando abaixo para mostrar as informações de cada campo da tabela `tbAlunos`.

DESCRIBE tbAlunos;

Field	Type	Null	Key	Default
idAluno	int(11)	NO		NULL
nomeAluno	varchar(64)	NO		NULL
enderecoAluno	varchar(30)	NO		NULL
bairroAluno	varchar(20)	NO		NULL
telefoneAluno	varchar(15)	NO		NULL
emailAluno	varchar(20)	NO		NULL
idadeAluno	int(3)	NO		NULL

Adicionando um campo em uma tabela existente

Para acrescentar um novo campo de dados a uma tabela já criada, usamos o comando sql abaixo. Vamos acrescentar na tabela tbAlunos o campo dataNiver (Data de aniversário) após o campo idadeAluno. Veja a sintaxe;

```
ALTER TABLE <nome_da_tabela> ADD <nome_do_campo_novo> [tipo] AFTER
<nome_do_campo_existente>;
```

```
ALTER TABLE tbAlunos ADD dataNiver date NOT NULL AFTER idadeAluno;
```

Alterando um campo em uma tabela existente

Após ter criado uma tabela com todos os seus campos já definidos, pode surgir a necessidade de renomear um dos campos. Veja a sintaxe;

```
ALTER TABLE <nome_da_tabela> CHANGE <nome_antigo> <nome_novo> [tipo];
```

Vamos alterar o campo telefoneAluno da tabela tbAlunos para telAlunos.

```
ALTER TABLE tbAlunos CHANGE telefoneAluno telAluno VARCHAR(18) NOT NULL;
```

Deletando uma coluna em uma tabela existente

Para deletar um dos campos de uma tabela, um dos comandos de alteração de tabela é o `DROP` que exclui o campo especificado. Veja a sintaxe;

```
ALTER TABLE <nome_da_tabela> DROP <nome_do_campo_a_ser_excluido>
```

```
ALTER TABLE tbAlunos DROP bairroAluno;
```

Deletando uma tabela existente

Sintaxe;

```
DROP TABLE <nome_da_tabela_a_ser_excluida>;
```

Crie uma tabela nova com o nome “tabalaTeste” somente para poder em seguida aplicar o comando `DROP TABLE`.

```
DROP TABLE tabelaTeste;
```


2- Na tabela tbProfessores que você criou anteriormente, descreva o comando SQL que informa as propriedades da tabela;

Aula 4 – Instrução Select

A instrução `select` é a mais importante do SQL, porque a partir dela é possível entender certas outras instruções semelhantes da linguagem SQL.

O comando `select` realiza vários tipos de consultas ao banco de dados retornando resultados gerais ou específicos em forma de tabelas.

Tabelas de exemplos para consultas

A partir desta aula e nas demais seguintes, vamos fazer várias consultas a um banco de dados de exemplos. Para poder realizar os vários comandos pretendidos para exemplificar o uso do comando `select` e suas variações, precisamos de algumas tabelas criadas e já preenchidas com uma determinada quantidade de registros. Portanto, vamos ter que criar as tabelas exibidas abaixo e inserir os dados listados em seguida.

Nome do banco: dbEscola

```
CREATE DATABASE dbEscola
```

Nome da Tabela: tbAlunos

```
CREATE TABLE tbAlunos (
  idAluno INT(10) NOT NULL PRIMARY KEY AUTO_INCREMENT,
  nomeAluno VARCHAR(45) NOT NULL,
  enderecoAluno VARCHAR(45) NOT NULL,
  idadeAluno INT(2) NOT NULL,
  cidadeAluno VARCHAR(45) NOT NULL,
  telefoneAluno VARCHAR(15) NOT NULL,
  idCurso INT(10) NOT NULL
)
```

Registros para popular a tabela:

idAluno	nomeAluno	enderecoAluno	cidadeAluno	idadeAluno	telefoneAluno	idCurso
1	João Alencar	Rua: Carmen Sandiego, 555	Campinas	17	(19) 5555-8888	1
2	Manuel Brito	Av. Amoreiras, 9063	Sumaré	20	(19) 3333-4545	1
3	Feliciano Souza	Rua: Andrade Neves, 569	Hortolândia	21	(19) 7878-3434	2
4	Luiza Aparecida	Rua: Mangones Silva, 458	Campinas	18	(21) 2325-5656	1
5	Júlio Verissimo	Av. Dr. Campos Salles, 405	Sumaré	19	(11) 9898-4848	3
6	Nicolý Melo	Rua: Bruno Vespertino, 684	Campinas	15	(11) 5654-7896	1

Nome da Tabela: tbFuncionarios

```
CREATE TABLE tbFuncionarios(
  idFuncionario INT(10) NOT NULL PRIMARY KEY AUTO_INCREMENT,
  nomeFuncionario VARCHAR(45) NOT NULL,
  enderecoFuncionario VARCHAR(45) NOT NULL,
  telefoneFuncionario VARCHAR(15),
  areaFuncionario VARCHAR(30),
  idCargo INT(10) NOT NULL,
  salarioFuncionario DECIMAL(10,2) NOT NULL
)
```

Registros para popular a tabela:

idFuncionario	nomeFuncionario	enderecoFuncionario	telefoneFuncionario	areaFuncionario	idCargo	salarioFuncionario
1	Bernardo Castro	Rua: Gusman Souza, 16	(21) 2325-5656	Administração	1	4500,00
2	Francisco Chaves	Av. Amorim Filho, 8863	(19) 3333-5757	Técnico	3	2500,00
3	Zangiev Victor	Rua: Glicério, 333	(11) 1549-1536	Coordenação	2	950,00
4	Deric Michael	Rua: Mangones Silva, 458	(19) 3854-3956	Técnico	3	1900,00
5	Alnir Klein	Av. Dr. Campos Salles, 6945	(11) 9898-8484	Técnico	3	1500,00
6	Franck Miller	Rua: Dr. Durval Miranda, 99	(19) 8184-5263	Coordenação	2	899,00
7	Marcos de Melo	Rua: Violeta, 301	(19) 8185-3695	Coordenação	2	3500,00

Nome da Tabela: tbCursos

```
CREATE TABLE tbCursos(
  idCurso INT(10) NOT NULL PRIMARY KEY AUTO_INCREMENT,
  nomeCurso VARCHAR(45) NOT NULL,
  valorCurso DECIMAL(10,2) NOT NULL
)
```

Registros para popular a tabela:

idCurso	nomeCurso	valorCurso
1	Matemática	150.00
2	Web Designer	250.00
3	Hardware	300.00
4	Inglês	90.00

Nome da Tabela: tbCargo

```
CREATE TABLE tbCargo(
  idCargo INT(10) NOT NULL PRIMARY KEY AUTO_INCREMENT,
```

```
descricaoCargo VARCHAR(45) NOT NULL
```

```
)
```

Registros para popular a tabela:

idCargo	descricaoCargo
1	Diretor
2	Coordenador
3	Professor

Comando básico do select

A sintaxe da instrução `select` mostrada abaixo exibe um ou mais campos especificados de um registro e como resultado, é mostrado a quantidade total de registros contidos no banco. Isso porque, não foi especificado nenhuma condição para a consulta.

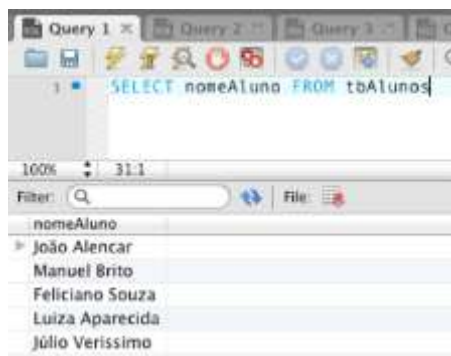
Sintaxe;

```
SELECT <campos, ..> FROM <nome_da_tabela>
```

Vamos aplicar uma consulta sql para mostrar o campo `nomeAluno` de todos os registros contidos na tabela `tbAlunos` do banco `dbEscola` criado para ser usado como exemplo para esta aula. Com o banco de dados `dbEscola` em uso, digite o comando SQL abaixo no programa Workbench para realizar esta consulta;

```
SELECT nomeAluno FROM tbAlunos
```

Resultado:

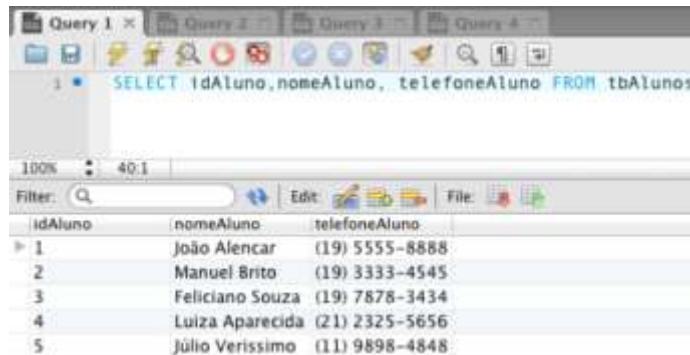


Repare que, como resultado, somente a coluna do campo `nomeAluno` foi visualizada. Isto porque, pedimos para mostrar somente este campo. É possível mostrar mais campos separados por vírgula (,).

Digite o código abaixo para mostrar todos os registros, retornando os campos idAluno, nomeAluno e telefoneAluno;

```
SELECT idAluno, nomeAluno, telefoneAluno FROM tbAlunos
```

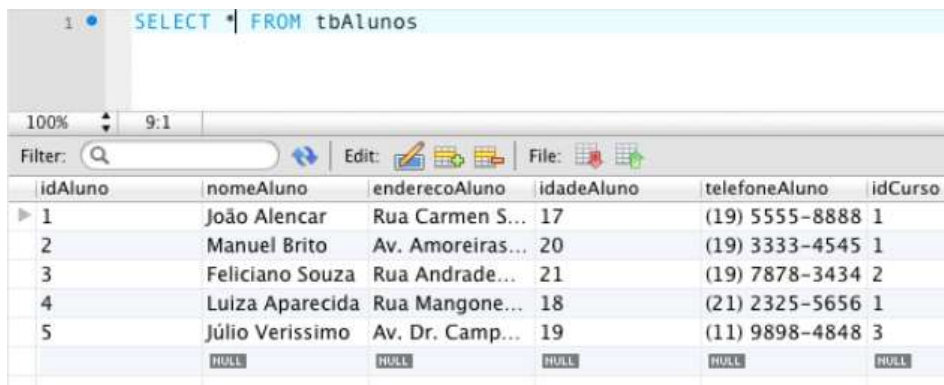
Resultado:



idAluno	nomeAluno	telefoneAluno
1	João Alencar	(19) 5555-8888
2	Manuel Brito	(19) 3333-4545
3	Feliciano Souza	(19) 7878-3434
4	Luiza Aparecida	(21) 2325-5656
5	Júlio Verissimo	(11) 9898-4848

Caso a necessidade seja, mostrar todos os campos de cada registro retornado, não precisamos indicar todos os campos, aliás, não precisamos indicar nenhum, basta usar o caractere coringa asterisco “ * “, para visualizar todos os campos na consulta.

```
SELECT * FROM tbAlunos
```



idAluno	nomeAluno	enderecoAluno	idadeAluno	telefoneAluno	idCurso
1	João Alencar	Rua Carmen S...	17	(19) 5555-8888	1
2	Manuel Brito	Av. Amoreiras...	20	(19) 3333-4545	1
3	Feliciano Souza	Rua Andrade...	21	(19) 7878-3434	2
4	Luiza Aparecida	Rua Mangone...	18	(21) 2325-5656	1
5	Júlio Verissimo	Av. Dr. Camp...	19	(11) 9898-4848	3

Atividades

1 – Descreva abaixo o comando para consultar e visualizar os campos **idFuncionario**, **nomeFuncionario** de cada registro na tabela **tbFuncionarios**.

R:

2 – Descreva abaixo o comando para consultar e visualizar todos os campos de cada registro na tabela **tbCursos**.

R:

Aula 5 - Cláusula Where / Operadores de Comparação

A Cláusula `Where` é um parâmetro opcional das instruções `select`, `update`, `delete`. É usada quando, queremos filtrar registros específicos de um todo, definindo condições para este filtro, onde valores de determinados campos, são comparados com outros e caso a condição seja verdadeira, o registro avaliado será selecionado.

Sintaxe;

```
SELECT * FROM <nome_da_tabela> WHERE <condição>
```

A cláusula `WHERE` usa como parâmetro de comparação, os operadores de comparação e os operadores SQL especiais, que vamos compreender mais adiante;

Operadores de comparação

Abaixo podemos ver a lista de operadores de comparação compreendidos pelo MySQL. Vamos ver cada um dos operadores explicados separadamente.

OPERADOR	SIGNIFICADO
=	Igual a
!=	Diferente
<>	
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a

Igual a “ = ”

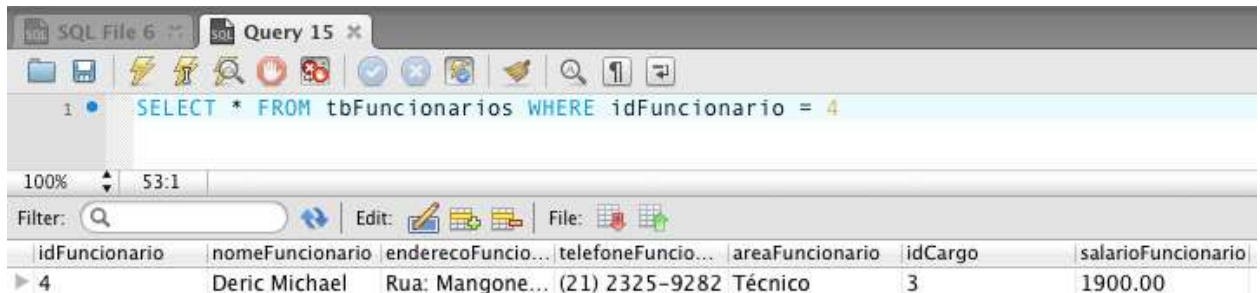
Este operador compara a igualdade de valores de um determinado campo com uma informação descrita.

Digite o código abaixo para que possamos selecionar somente o registro cujo `idFuncionario` seja igual a o número 4 na tabela `tbFuncionarios`.

Exemplo;

```
SELECT * FROM tbFuncionarios WHERE idFuncionario = 4
```

Resultado:



The screenshot shows a MySQL Query Editor window with the following query: `SELECT * FROM tbFuncionarios WHERE idFuncionario = 4`. The result set is displayed in a table below the query editor.

idFuncionario	nomeFuncionario	enderecoFuncio...	telefoneFuncio...	areaFuncionario	idCargo	salarioFuncionario
4	Deric Michael	Rua: Mangone...	(21) 2325-9282	Técnico	3	1900.00

Diferente “ != “ ou “ <> ”

O operador “**Diferente**” proporciona realizar a ação contrária ao “**Igual a**”. Ou seja, a consulta retornara todos os registros que não tiverem no campo especificado, o valor indicado no parâmetro.

No exemplo abaixo todos os registros de cursos cujo o campo idCurso seja diferente de 3 serão retornados na consulta.

Exemplo;

```
SELECT * FROM tbCursos WHERE idCurso !=3
```

Resultado:



The screenshot shows a MySQL Query Editor window with the following query: `SELECT * FROM tbCursos WHERE idCurso != 3`. The result set is displayed in a table below the query editor.

idCurso	nomeCurso	valorCurso
1	Informática	150.00
2	Web Designer	250.00

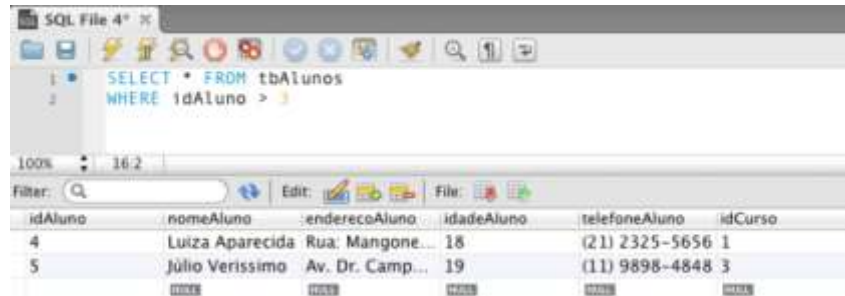
Maior que “ > ”

Selecione os registros maiores que o valor especificado na condição.

Exemplo:

```
SELECT * FROM tbAlunos WHERE idAluno >3
```

Resultado:



idAluno	nomeAluno	enderecoAluno	idadeAluno	telefoneAluno	idCurso
4	Luiza Aparecida	Rua: Mangone...	18	(21) 2325-5656	1
5	Júlio Veríssimo	Av. Dr. Camp...	19	(11) 9898-4848	3

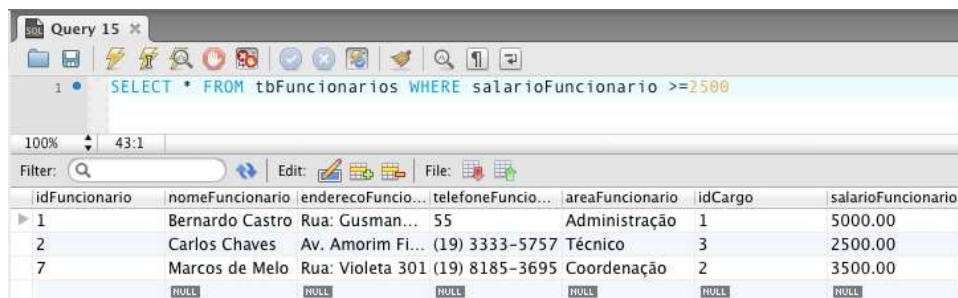
Maior ou igual a “>= ”

Seleciona os registros maiores ou iguais ao valor especificado na condição.

Exemplo:

```
SELECT * FROM tbFuncionarios WHERE salarioFuncionario >= 2500
```

Resultado:



idFuncionario	nomeFuncionario	enderecoFuncio...	telefoneFuncio...	areaFuncionario	idCargo	salarioFuncionario
1	Bernardo Castro	Rua: Gusman...	55	Administração	1	5000.00
2	Carlos Chaves	Av. Amorim Fi...	(19) 3333-5757	Técnico	3	2500.00
7	Marcos de Melo	Rua: Violeta 301	(19) 8185-3695	Coordenação	2	3500.00
	NULL	NULL	NULL	NULL	NULL	NULL

Menor que “< ”

Seleciona os registros menores que o valor especificado na condição.

Exemplo:

```
SELECT * FROM tbFuncionarios WHERE salarioFuncionario < 2500
```

Resultado:

Query 15 x

```
1 • SELECT * FROM tbFuncionarios WHERE salarioFuncionario <2500
```

100% 56:1

Filter: Edit: File:

idFuncionario	nomeFuncionario	enderecoFuncio...	telefoneFuncio...	areaFuncionario	idCargo	salarioFuncionario
3	Zangiev Victor	Rua: Andrade...		Coordenação	2	950.00
4	Deric Michael	Rua: Mangone...	(21) 2325-9282	Técnico	3	1900.00
5	Alnir Klein	Av. Dr. Camp...	(11) 9898-8484	Técnico	3	1500.00
6	Franck Miller	Rua: Dr. Durv...	(11) 4569-4975	Coordenação	2	899.00

Menor ou igual a “<=”

Seleciona os registros menores ou iguais ao valor especificado na condição.

Exemplo:

```
SELECT * FROM tbAlunos WHERE idAluno <= 2
```

Resultado:

```
1 • SELECT * FROM tbAlunos WHERE idAluno <=2
```

100% 40:1

Filter: Edit: File:

idAluno	nomeAluno	enderecoAluno	idadeAluno	telefoneAluno	idCurso
▶ 1	João Alencar	Rua: Carmen...	17	(19) 5555-8888	1
2	Manuel Brito	Av. Amoreiras...	20	(19) 3333-4545	1

Atividades

1 – Descreva abaixo o comando para consultar e visualizar os campos **idFuncionario**, **nomeFuncionario** de cada registro na tabela **tbFuncionarios** cujo o campo **salarioFuncionario** seja maior ou igual a **1500**.

R:

2 – Descreva abaixo o comando para consultar e visualizar os campos **idAluno**, **nomeAluno** e **idadeAluno** de cada registro na tabela **tbAluno** cujo o campo **idadeAluno** seja menor ou igual a **18**.

R:

Aula 6 – Cláusula Where / Comandos Especiais SQL

Vamos abordar nesta aula os comandos especiais SQL referentes a cláusula Where.

Operadores SQL especiais

O SQL possui quatro operadores especiais:

OPERADOR	SIGNIFICADO
BETWEEN NOT BETWEEN	Intervalo fechado compreendido entre os dois valores val1 e val2
IN (val1,val2,val3,val4) NOT IN (val1,val2,val3,val4)	Pertence à lista de valores val1, val2, val3, val4
LIKE NOT LIKE	Cadeia de caracteres que satisfaz a condição. Utilizar % e _
IS NULL IS NOT NULL	É um valor NULL

BETWEEN

Retorna valores consultados compreendidos entre um valor mínimo e máximo. Utilizado na construção de condições, por exemplo para cláusula WHERE.

Sintaxe;

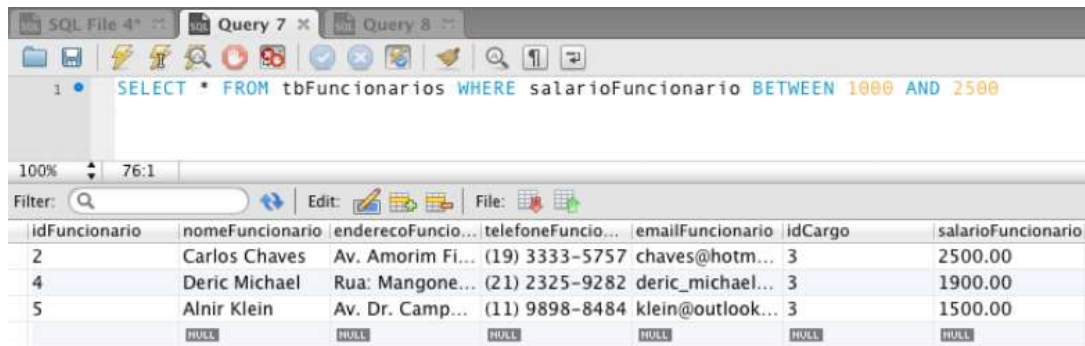
```
SELECT <campo> FROM <nome_da_tabela> WHERE <campo> BETWEEN <valor_minimo> AND <valor_maximo>
```

Exemplo:

Neste exemplo, a consulta retorna todos os registros cujo os valores no campo salarioFuncionario esteja no intervalo de 1000 à 2000.

```
SELECT * FROM tbFuncionarios WHERE salarioFuncionario BETWEEN 1000 AND 2500
```

Resultado:



Query 7: `SELECT * FROM tbFuncionarios WHERE salarioFuncionario BETWEEN 1000 AND 2500`

idFuncionario	nomeFuncionario	enderecoFuncio...	telefoneFuncio...	emailFuncionario	idCargo	salarioFuncionario
2	Carlos Chaves	Av. Amorim Fi...	(19) 3333-5757	chaves@hotm...	3	2500.00
4	Deric Michael	Rua: Mangone...	(21) 2325-9282	deric_michael...	3	1900.00
5	Alnir Klein	Av. Dr. Camp...	(11) 9898-8484	klein@outlook...	3	1500.00

IN (val1, val2, val3, val4)

Procura os valores numa lista especificada.

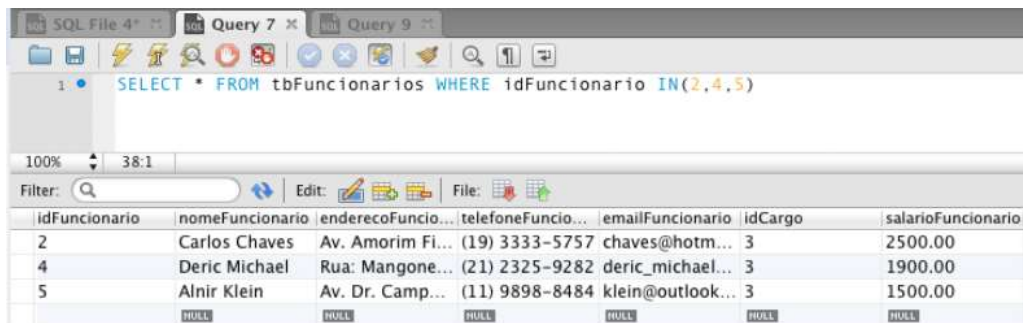
Sintaxe;

```
SELECT <campo> FROM <nome_da_tabela> WHERE <campo> IN(<valor1>, <valor2>,..)
```

Exemplo:

```
SELECT * FROM tbFuncionario WHERE idFuncionario IN(2,4,5)
```

Resultado:



Query 9: `SELECT * FROM tbFuncionarios WHERE idFuncionario IN(2,4,5)`

idFuncionario	nomeFuncionario	enderecoFuncio...	telefoneFuncio...	emailFuncionario	idCargo	salarioFuncionario
2	Carlos Chaves	Av. Amorim Fi...	(19) 3333-5757	chaves@hotm...	3	2500.00
4	Deric Michael	Rua: Mangone...	(21) 2325-9282	deric_michael...	3	1900.00
5	Alnir Klein	Av. Dr. Camp...	(11) 9898-8484	klein@outlook...	3	1500.00

LIKE

Quando não se conhece o valor exato a procurar, mas temos uma ideia aproximada, podemos utilizar o operador LIKE. Permite selecionar linhas que concordem com um dado padrão de caracteres. A cadeia de caracteres usada como padrão de pesquisa pode utilizar dois símbolos especiais:

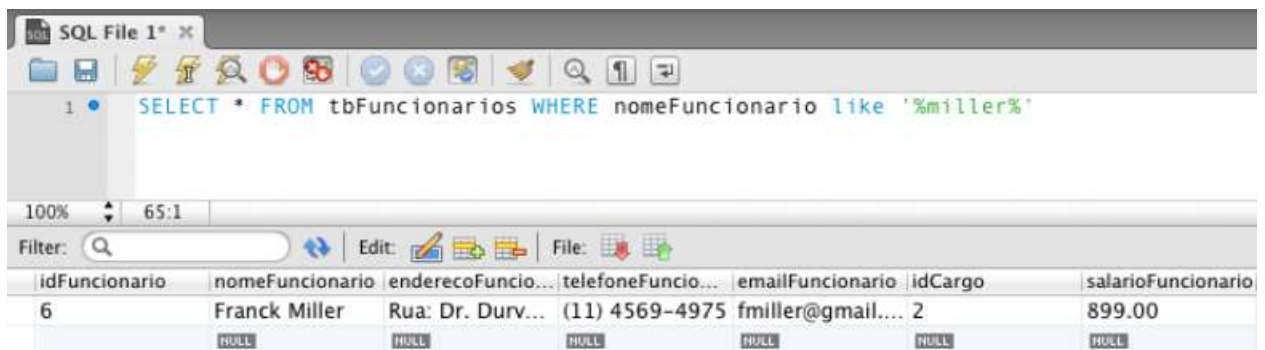
SIMBOLO	REPRESENTA
%	Qualquer cadeia com nenhum ou vários caracteres
_	Um caractere qualquer

SELECT <campo> FROM <nome_da_tabela> WHERE <campo> LIKE '%<string>%';

Exemplo:

```
SELECT * FROM tbFuncionarios WHERE nomeFuncionario LIKE '%miller%';
```

Resultado:



IS NULL

Os operadores de comparação IS NULL e IS NOT NULL são usados quando precisamos verificar se determinados campos de nossas tabelas MySQL contém valores NULL.

Tenha em mente que NULL é diferente de 0 (zero) e uma String vazia, ou seja, campos com valores NULL são aqueles que não possuem nenhum valor (NULL significa ausência de valor). O operador IS NOT NULL testa se o valor de um determinado campo não é NULL.

Sintaxe:

```
SELECT <campo> FROM <nome_da_tabela> WHERE <campo> IS NULL
```

Atividades

1 – Descreva abaixo o comando SQL para visualizar na tabela **tbFuncionarios**, somente os registros cujo o campo **idFuncionario** seja 2,3,5,7.

R:

2 – Descreva abaixo o comando SQL para visualizar na tabela **tbFuncionarios**, somente os registros cujo os valores no campo **idFuncionario** estejam no intervalo de 3 a 6.

R:

Aula 7 –Parâmetros da instrução Select / Parte 1

A partir deste aula vamos abordar mais alguns parâmetros da instrução **select**. A compreensão destes parâmetros está dividida em duas partes entre a aula 7 e 8.

Parâmetro JOIN

O JOIN é um dos mais importantes comando da linguagem SQL, pois tem a capacidade de retornar dados contidos em mais de uma tabela, através de campos relacionados.

Sintaxe;

```
SELECT <campo> FROM <tabela_1> JOIN <tabela_2>  
ON <tabela_1>.<campo>=<tabela_2>.<campo>  
WHERE <condições>
```

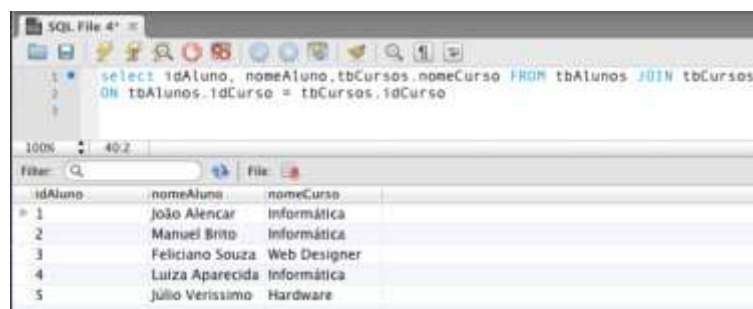
Exemplo:

A tabela de Alunos possui um campo chamado idCurso cuja a informação é relacionada com o código de chave primaria em tbCursos, identificando assim, o curso de cada aluno.

No exemplo a seguir seleciona todos os registros exibindo o idAluno, nomeAluno e também o nomeCurso relacionado na com a tabela tbCurso, usando o parâmetro JOIN.

```
SELECT idAluno,nomeAluno,nomeCurso FROM tbAlunos JOIN tbCursos  
ON tbAlunos.idCurso=tbCursos.idCurso
```

Resultado:



idAluno	nomeAluno	nomeCurso
1	João Alencar	Informática
2	Manuel Brito	Informática
3	Feliciano Souza	Web Designer
4	Luiza Aparecida	Informática
5	Júlio Verissimo	Hardware

Parâmetro ORDER BY

O parâmetro `ORDER BY` é utilizado na instrução `SELECT` para ordenar uma coluna de dados em ordem Crescente ou Decrescente ou também do maior valor numérico para o menor e vice versa com base no campo(coluna) especificado.

Sintaxe;

```
SELECT <campo> FROM <nome_da_tabela> ORDER BY <campo> [ASC|DESC]
```

Os parâmetros `[ASC|DESC]` são opcionais.

ASC - Ordena os resultados do campo/coluna em ordem crescente (A-Z).

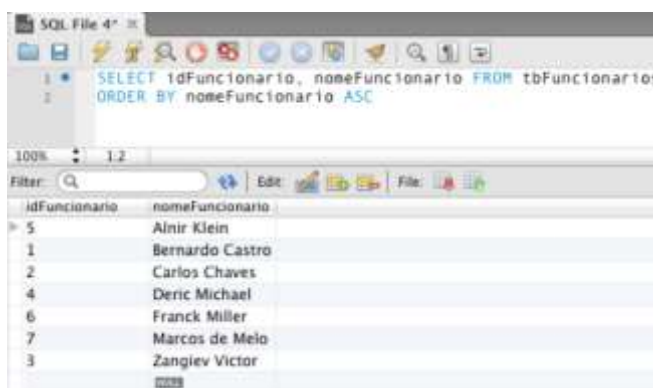
DESC - Ordena os resultados do campo/coluna em ordem decrescente (Z-A).

Exemplo;

Vamos selecionar todos os dados da tabela `tbFuncionarios` no campo `nomeFuncionario` em ordem Crescente "A-Z".

```
SELECT idFuncionario, nomeFuncionario FROM tbFuncionarios
```

```
ORDER BY nomeFuncionario ASC
```



idFuncionario	nomeFuncionario
5	Almir Klein
1	Bernardo Castro
2	Carlos Chaves
4	Deric Michael
6	Franck Miller
7	Marcos de Melo
3	Zangiev Victor

Parâmetro DISTINCT(Omite registros duplicados)

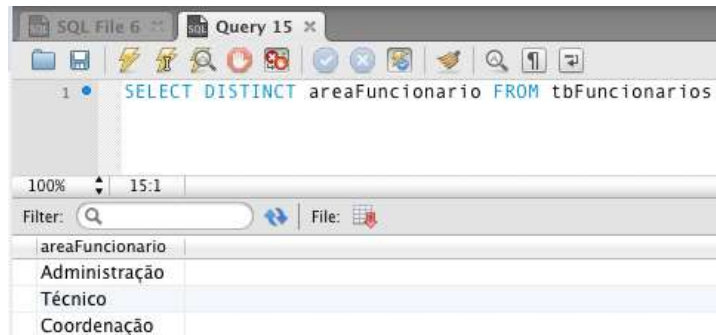
O parâmetro `DISTINCT` elimina linhas repetidas considerando todas as colunas como um todo.

Sintaxe;

```
SELECT DISTINCT <campo> FROM <nome_da_tabela>
```

Exemplo:

SELECT DISTINCT areaFuncionario FROM tbFuncionarios



Parâmetro LIMIT (limitador de registros selecionados)

O parâmetro LIMIT é usado para limitar a quantidade de resultados em uma consulta de seleção. Através do comando LIMIT é possível extrair dados como os 5 primeiros ou 5 últimos registros de uma tabela. Outra utilidade para o parâmetro LIMIT é a paginação de resultados nas consultas.

Sintaxe:

```
SELECT <campo> FROM <nome_da_tabela> LIMIT X,Y
```

X – valor numérico que indica a posição inicial a partir de onde os registros serão exibidos. Registros antes desta posição serão ignorados.

Y – Valor numérico que indica a quantidade de registros há serem exibidos a partir da posição X.

Observações:

Em uma tabela de registros a posição 0 (zero) corresponde a primeira linha de registros, portanto, quando definimos, por exemplo, a posição 9 no parâmetro LIMIT, estaremos selecionando o decimo registro da tabela.

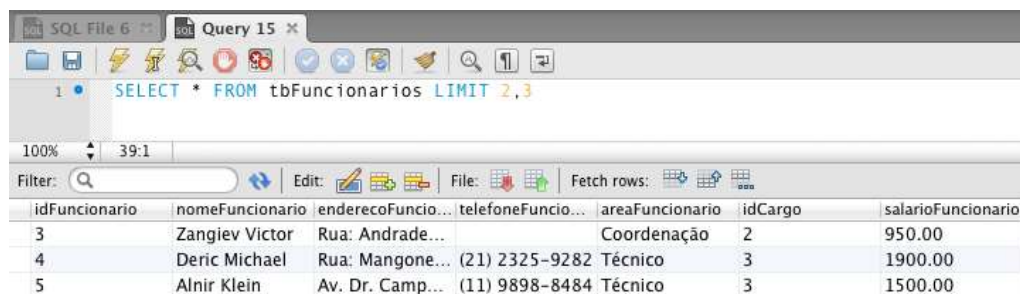
Índices	Registros por linhas
0	1º
1	2º
2	3º
3	4º
4	5º
5	6º
6	7º
7	8º
8	9º
9	10º

Exemplo:

O comando SQL abaixo usa o parâmetro LIMIT para mostra a partir da posição 2 na tabela do banco de dados tbFuncionarios a quantidade de 3 registros somente.

SELECT idFuncionario, nomeFuncionario FROM tbFuncionarios LIMIT 2,3

Resultado:



idFuncionario	nomeFuncionario	enderecoFuncio...	telefoneFuncio...	areaFuncionario	idCargo	salarioFuncionario
3	Zangiev Victor	Rua: Andrade...		Coordenação	2	950.00
4	Deric Michael	Rua: Mangone...	(21) 2325-9282	Técnico	3	1900.00
5	Alnir Klein	Av. Dr. Camp...	(11) 9898-8484	Técnico	3	1500.00

Parâmetro COUNT (Contagem de registros)

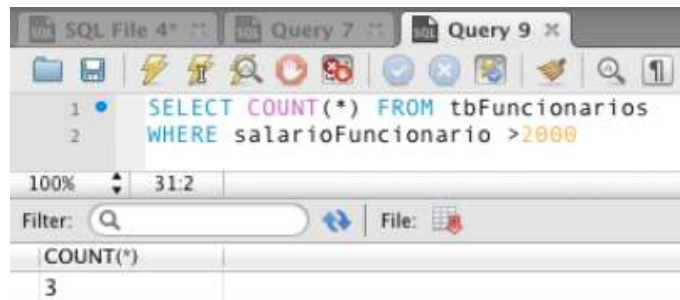
O parâmetro COUNT realiza a contagem de registros selecionados retornando um valor numérico com a totalização da contagem. Podemos aplicar um filtro pela clausula WHERE para condicionar quais registros serão contados.

Sintaxe;

```
SELECT COUNT( <campo>) FROM <nome_da_tabela> WHERE <condição>
```

Exemplo:

```
SELECT COUNT(*) FROM tbFuncionarios WHERE salarioFuncionario > 2000
```



Aula 8 –Parâmetros da instrução Select / Parte 2

Nesta aula continuaremos o entendimento dos parâmetros associados à instrução SELECT.

Parâmetro SUM (Soma de valores)

O parâmetro SUM realiza a somatória de campos por registro selecionado, retornando um valor numérico com o resultado do cálculo da soma. Podemos aplicar um filtro pela cláusula WHERE para condicionar quais registros terão campos de valores numéricos somados.

Sintaxe;

```
SELECT SUM(<campo>) FROM <nome_da_tabela> WHERE <condição>
```

Exemplo:

```
SELECT SUM(salarioFuncionario) FROM tbFuncionarios WHERE  
idCargo = 2
```



Parâmetro AVG (Média de valores)

O parâmetro AVG calcula a média de valores, de campos com valores numéricos em uma consulta SQL. Podemos utilizar a cláusula WHERE para filtrar as linhas de registros que serão calculadas.

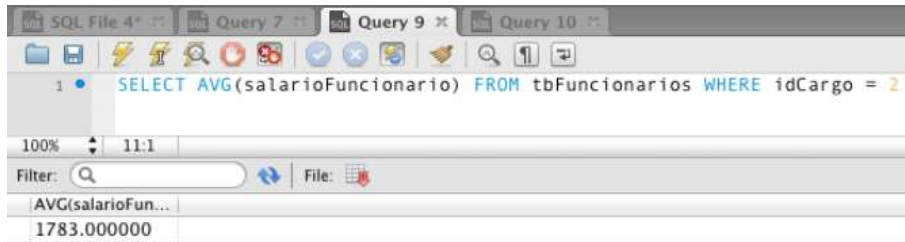
Sintaxe;

```
SELECT AVG( <campo> ) FROM <nome_da_tabela> WHERE <condição>
```

Exemplo:

No exemplo a seguir é calculado a média de valores do campo `salarioFuncionario` de todos os funcionários cujo campo `idCargo` seja igual a 2.

```
SELECT AVG(salarioFuncionario) FROM tbFuncionarios WHERE idCargo = 2
```

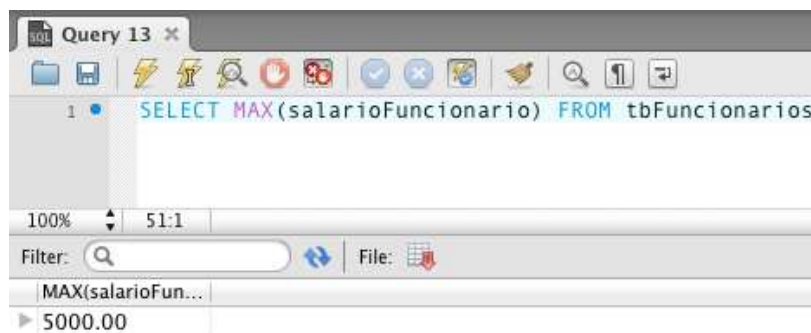


Parâmetro MAX/MIN

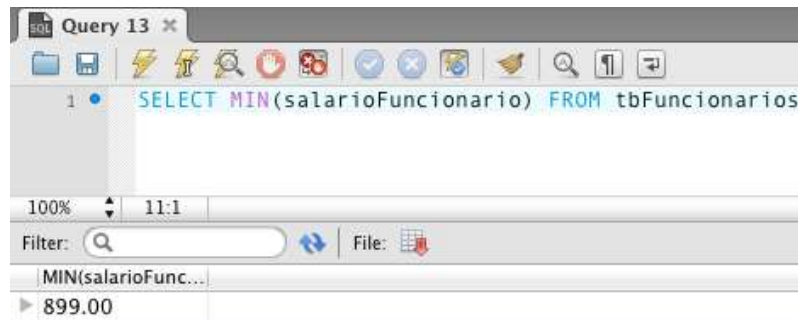
Os parâmetros MAX e MIN são utilizados para se obter o maior ou menor valor de um campo numa consulta SQL.

Exemplo:

Na consulta SQL a seguir, utilizando o parâmetro `MAX` no campo `salarioFuncionario` será retornado o valor do maior salário pago a funcionários.



Na consulta SQL a seguir, utilizando o parâmetro `MIN` no campo `salarioFuncionario` será retornado o valor do menor salário pago a funcionários.



Parâmetro GROUP BY

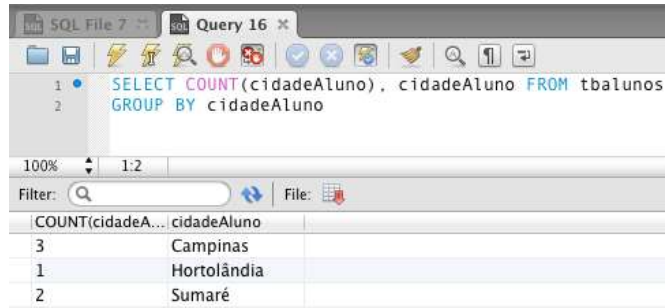
A cláusula GROUP BY é usada para agrupar as linhas da tabela segundo um critério definido usando a cláusula WHERE. Os grupos são formados pelo agrupamento das linhas, a cada grupo formado no passo anterior são aplicadas as funções de grupo (se houver).

Quando se usa GROUP BY estamos a trabalhar em "**modo de grupo**" o que implica deixar de considerar linhas individuais para considerar grupos de linhas. A cada grupo de linhas apenas podemos aplicar funções de grupo, tais como: média, variância, valor máximo, valor mínimo ou contagem do número de elementos.

Exemplo:

O exemplo abaixo mostra quantos Alunos são de cada cidade. As linhas da tabela tbAlunos são ordenadas pelo campo cidadeAluno. Depois são formados conjuntos de linhas, cujos elementos, têm em comum o valor de cidadeAluno. Finalmente é feita a contagem do número de linhas de cada conjunto:

```
SELECT COUNT(cidadeAluno), cidadeAluno FROM tbAlunos  
GROUP BY cidadeAluno
```



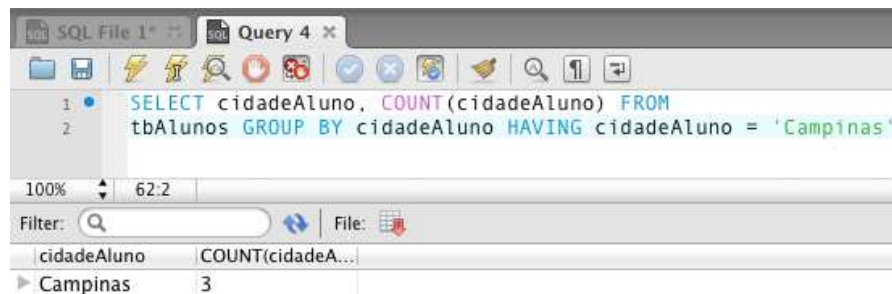
Parâmetro HAVING

O parâmetro HAVING é semelhante ao parâmetro condicional WHERE, mas a diferença é que o HAVING deve ser usado com o GROUP BY, pois sua consulta é baseada em colunas agrupadas.

Sintaxe;

Exemplo

```
SELECT cidadeAluno, COUNT(cidadeAluno) FROM
tbAlunos GROUP BY HAVING cidadeAluno = 'Campinas'
```



Aula 9 – Instruções de Inserções, atualizações e exclusões

Até agora focamos muito o uso a instrução SELECT e seus parâmetros. Nesta aula vamos compreender o uso de três instruções principais da linguagem SQL, as instruções; INSERT, UPDATE e DELETE.

Insert

A instrução INSERT não tem muita complexidade, basicamente é um comando padrão de inserção de novos valores em uma tabela de banco de dados.

Sintaxe;

```
INSERT INTO <nome_da_tabela> (<campo1>,<campo2>,...)
VALUES ( <valor_campo1> , <valor_campo2>,...)
```

Exemplo:

```
INSERT INTO tbCurso (nomeCurso, valorCurso) VALUES ('Português' , 100.00)
```

Update

A instrução UPDATE é utilizada para atualizar valores em determinados campos de um registro ou de vários registros ao mesmo tempo. A instrução UPDATE deve ser usada cuidadosamente com a clausula WHERE, pois sem a clausula where todas as linhas de registros serão atualizados.

Uma instrução UPDATE permite alterar:

- Uma coluna de uma linha da tabela;
- Várias colunas de uma linha;
- Uma coluna em várias linhas;
- Várias colunas em várias linhas;

Para obter o que foi descrito acima usamos:

- A cláusula SET para escolher a(s) coluna(s);

- A cláusula WHERE para escolher a(s) linha(s);

Sintaxe;

```
UPDATE <nome_da_tabela> SET <campo1> = <valor1>, <campo2> = <valor2>, ...  
WHERE <condição>
```

Exemplo 1:

No exemplo abaixo, o campo `salarioFuncionario` é atualizado para 5000, mas com a condição imposta pela cláusula WHERE que atualiza somente o registro do funcionário cujo o campo `idFuncionario` seja igual a 1.

```
UPDATE tbFuncionarios SET salarioFuncionario = 5000 WHERE idFuncionario = 1
```



Após a execução da consulta UPDATE, o funcionário Bernardo Castro terá seu salário atualizado para 5000,00.

Observações:

Caso a cláusula WHERE não fosse informada a atualização do campo `salarioFuncionario` aconteceria em todos os registros de funcionários.

Exemplo 2:

É possível também, atualizar dados numéricos de um campo em cada registro selecionado para o resultado de um cálculo matemático em tempo de execução.

No exemplo a seguir, querem acrescentar R\$ 100,00 reais de bonificação ao salário de todos os funcionários. Para realizar esta consulta Update, adicionaremos na consulta SQL um pequeno cálculo a seguir.

```
UPDATE tbFuncionarios SET salarioFuncionario = salarioFuncionario + 100
```

Entendendo claramente o que acontece neste cálculo;

Vamos supor que o campo salário já tenha 5000, então a atualização seria assim.

salarioFuncionario = salarioFuncionario + 100
--

salarioFuncionario = 5000+100

Delete

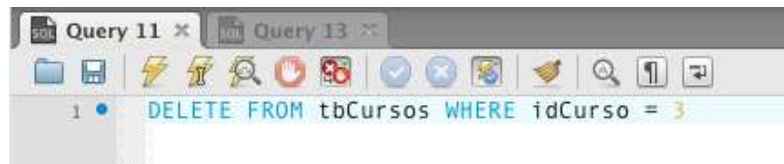
A instrução DELETE exclui linhas de registros em uma tabela. Aplicando junto com a cláusula WHERE podemos especificar qual ou quais linhas de registros serão excluídas.

Sintaxe;

```
DELETE FROM <nome_da_tabela> WHERE <condição>
```

Exemplo:

```
DELETE FROM tbCursos WHERE idCurso = 3
```



Atividades

1. Utilizando a instrução INSERT cadastre mais um aluno na tabela **tbAlunos** do banco de dados **dbEscola**.

R:

2. Utilizando a instrução UPDATE, atualize o campo **salarioFuncionario** na tabela **tbFuncionarios** para 5000, mas utilize a cláusula **WHERE** para especificar somente os funcionários cujo **idCargo** seja 3.

R:

3. Utilizando a instrução DELETE na tabela **tbCursos**, exclua o curso “Matemática”.

R:

Aula 10 – Controle de Usuários / Backups e Recovers

Vamos dedicar esta aula a compreensão do controle de acesso por usuários e a como realizar backups de segurança, bem como a restauração destes backups.

O banco de dados é a parte integrante do sistema que mais devemos dar importância, poderíamos dizer que o banco de dados é o coração de um sistema, seja ele, uma aplicação Web ou Desktop. Portanto, vários fatores de segurança devem ser levados em consideração, para garantir a segurança do banco de dados.

Podemos agrupar os fatores de segurança em dois tópicos principais; Controle de Usuários e Backups.

O primeiro fator importante, é com relação a segurança de acesso. Devemos definir as contas de usuários com suas respectivas senhas e definir os privilégios de acesso correto a cada usuário.

O segundo fator importante, é com relação à os backups e recovers, para garantir cópias fiéis dos bancos de dados de seus sistemas até a data que foi realizado o backup. Se o banco original apresentar problemas ou se você tenha, perdido dados acidentalmente, é possível restaurar o backup mais recente.

Controle de Usuários

Já sabemos que, para conectar a um servidor de banco de dados MySQL, seja ele, local o remoto, precisamos logar no mesmo, com uma conta de usuários. Esta conta de usuário precisa ter ainda, os privilégios, sejam eles, totais ou limitados, para ter acesso aos bancos de dados criados nestes servidor.

Criando um usuário

Para se criar um usuário é preciso antes de mais nada, estar logado no banco MySQL com um usuário que possua privilégios de criação e alteração de contas de usuários. Utilize o usuário “**root**” para realizar esta ação.

Sintaxe:

```
CREATE USER <usuário> [IDENTIFIED BY '<senha>']
```

No MySQL o nome do usuário é constituído de um nome mais o host de onde ele poderá acessar o servidor (user@host). Caso você não informe o host para o usuário, o MySQL assumirá "%", isto é, todos os hosts. A opção IDENTIFIED BY é para especificar a senha do usuário novo e é opcional, podendo ser configurado mais adiante.

Exemplo:

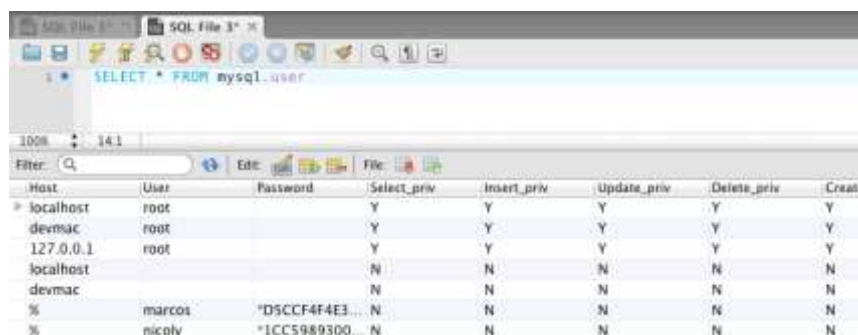
```
CREATE USER marcos IDENTIFIED BY 'senha123'
```



Exibindo usuários cadastrados

Todos os usuários são cadastrados em uma tabela interna do MySQL chamada "user" contida num banco de dados especial chamado 'mysql'. Para se exibir a lista de usuários de um banco de dados MySQL basta digitar a instrução abaixo;

```
SELECT * FROM mysql.user
```



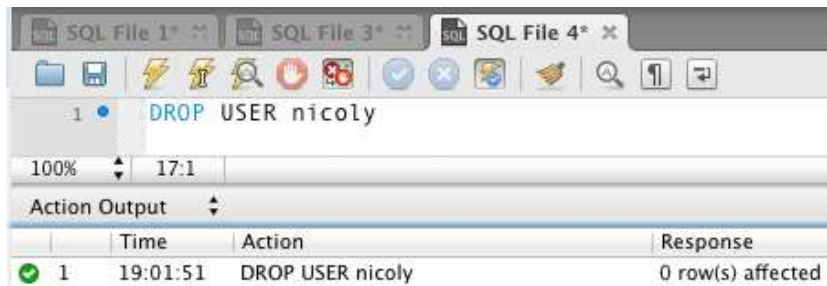
Excluindo um usuário

Para se excluir um usuário, novamente lembramos que deve-se estar logado como um usuário com privilégios, desta vez de exclusão de usuários.

O comando utilizada para excluir um usuário é o comando `DROP`, também utilizado para excluir bancos e tabelas.

Sintaxe;

DROP USER <nome_do_usuario>



Liberando Privilégios

O MySQL armazena as informações dos seus usuários em 4 tabelas que estão localizadas no banco de dados `mysql`. Estas tabelas são a `user`, `db`, `tables_priv` e `columns_priv`.

Tabela “user”

A tabela `user` armazena as informações dos usuários do banco e os privilégios globais deste usuário.

Tabela “tables_priv” e “columns_priv”

Armazena os privilégios dos usuários específicos de um banco de dados.

tables_priv e columns_priv

Armazenam os privilégios associados a tabelas e colunas, respectivamente.

Sintaxe;

```
GRANT <privilégios> [(colunas)] ON <banco>.<tabela> TO '<usuário>'@'<domínio>'
IDENTIFIED BY '<senha>'
```

No comando acima os [] indicam que o comando é opcional. O primeiro item a ser informado é(são) o(s) privilégio(s) a ser(em) concedido(s) ao(s) usuário(s).

O primeiro parâmetro após a instrução `GRANT <privilégios>` informa quais privilégios poderão ser atribuídos ao usuário indicado. Os privilégios podem ser colocados um a um, separados por vírgula(,).

Abaixo segue a lista de privilégios disponíveis para os usuários;

Privilégio	Descrição
ALL [PRIVILEGES]	Todos os privilégios exceto GRANT OPTION
ALTER	Permite executar alterações de tabelas (Editar)
CREATE	Permite a criação de novas tabelas
CREATE TEMPORARY TABLES	Permite a criação de tabelas temporárias
DELETE	Permite deletar dados nas tabelas
DROP	Permite executar DROP TABLE (Excluir tabelas)
EXECUTE	Permite executar stored procedures
FILE	Permite executar SELECT ... INTO OUTFILE e LOAD DATA INFILE
INDEX	Permite executar CREATE INDEX e DROP INDEX
INSERT	Permite executar a instrução INSERT nas tabelas
LOCK TABLES	Permite executar LOCK TABLES em tabelas que você tenha o privilégio SELECT
PROCESS	Permite executar SHOW FULL PROCESSLIST
REFERENCES	Ainda não está implementado
RELOAD	Permite executar FLUSH
REPLICATION CLIENT	Permite ao usuário obter a localização do Master ou Slave
REPLICATION SLAVE	Necessário para a replicação Slave (leitura dos eventos do log binário do Master)
SELECT	Permite executar SELECT
SHOW DATABASES	Exibe todos os bancos de dados
SHUTDOWN	Permite executar mysqladmin shutdown
SUPER	Permite executar CHANGE MASTER, KILL, PURGE MASTER LOGS e SET GLOBAL. Permite conectar-se ao servidor uma vez, mesmo que o max_connections tenha sido atingido
UPDATE	Permite executar a instrução UPDATE
USAGE	Sinônimo para "no privileges"
GRANT OPTION	Permite ao usuário repassar os seus privilégios

É possível definir as colunas que receberam este atributo, mas esta opção é opcional.

Após a definição dos privilégios, temos que definir os banco de dados e suas tabelas. Caso os privilégios devam ser aplicados a todos os bancos de dados e suas tabelas, podemos utilizar o caractere Asterisco (*) da seguinte forma;

```
GRANT ... ON *.* TO ....
```

Por fim, indicamos qual o usuário recebera estes privilégios.

```
.....TO `marcos`@'dominio.com.br'
```

Caso você queira especificar acesso por todos e quais quer domínios

Exemplo:

Neste exemplo a seguir, é concedido ao usuário “**marcos**” os privilégios descritos, em todos os bancos e todos as respectivas tabelas de cada um, através do parâmetro *.*

```
GRANT ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE VIEW,  
INDEX, SHOW DATABASES, SHOW VIEW  
ON *.* TO 'marcos'@'%'
```



Para listar os privilégios deste usuário utilize o comando:

```
SHOW GRANTS FOR marcos@localhost;
```

Backups e Recovers

Neste tópico vamos abordar os procedimentos de backups dos bancos de dados do MySQL, bem como a restauração desses backups.

Backup pelo phpMyAdmin

O programa phpMyAdmin é uma aplicação Web desenvolvida na linguagem PHP para manipular o banco de dados MySQL. Através do phpMyAdmin podemos fazer todo tipo de manipulação no banco de dados MySQL, desde que, o usuário que nos conectamos ao banco, tenha os privilégios de acesso para tudo o que vamos realizar no MySQL.

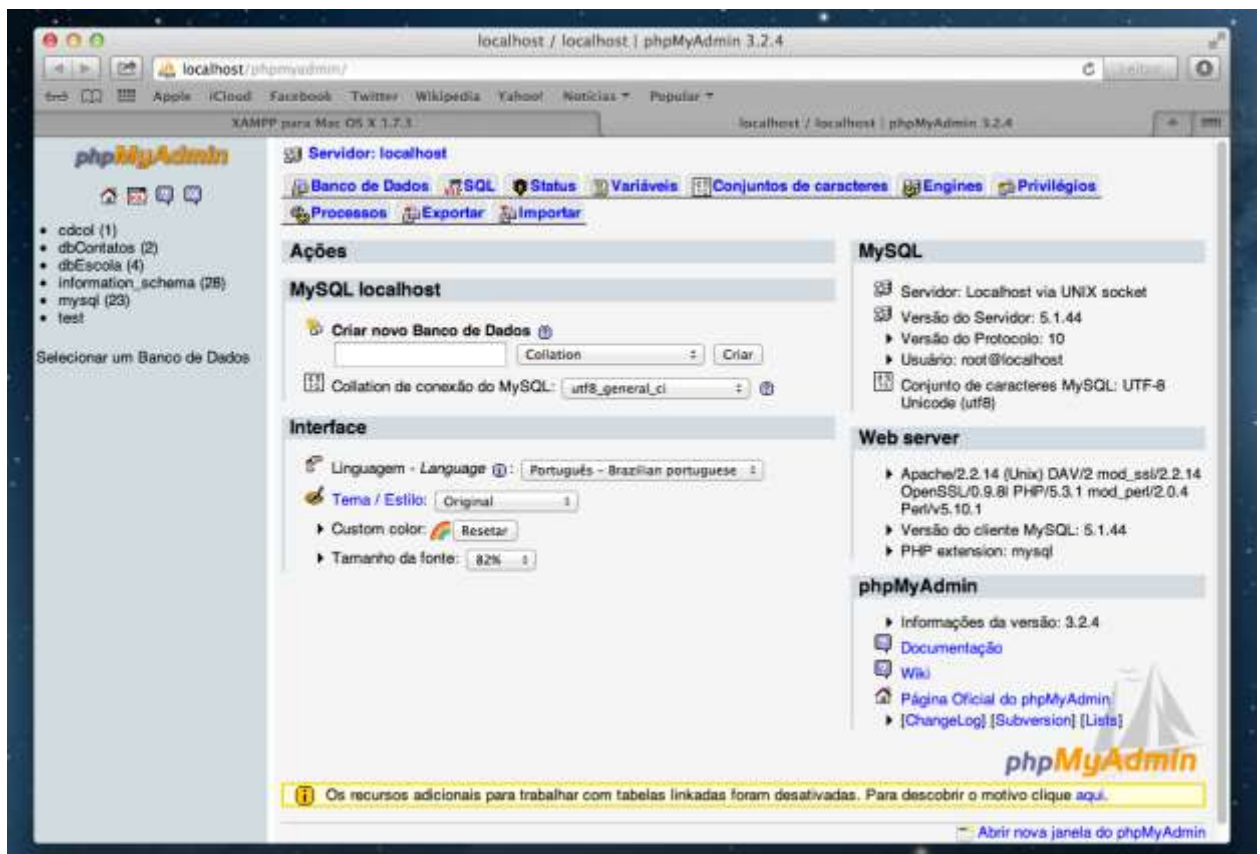
O phpMyAdmin é o cliente de acesso ao banco de dados MySQL mais usado pelos Servidores de hospedagem de sites com banco de dados MySQL. Geralmente o acesso a ele, é feito através de uma área de administração como, Painel de controle.

Além dos vários recursos de manipulação do banco de dados que podemos executar, podemos realizar backups e restauração dos mesmos pelo phpMyAdmin.

O XAMPP possui o phpMyAdmin instalado para acessar e manipular o banco de dados MySQL local e para acessa-lo, basta abrir o Navegador de internet (Browser) e digitar o link <http://localhost/phpmyadin/>.

Nota!

Lembre-se que o servidor apache deve estar ativo no painel de controle do xampp.



Criando um backup

1. Após acessar o phpMyAdmin, clique na guia Exportar para podermos acessar as configurações de criação de backup.
2. Na área de exportação, selecione os bancos de dados que deseja criar um backup (cópia de segurança).
3. Selecione a extensão **SQL** para o arquivo a ser gerando.
4. Defina o nome para o arquivo de backup.
5. Clique no botão **Executar** para gerar o backup.

Servidor: localhost

Banco de Dados SQL Status Variáveis Conjuntos de caracteres Engines Privilégios Processos Exportar

Importar

Ver dump (esquema) dos Bancos de Dados

Exportar

Selecionar Todos / Desmarcar Todos

- cdcol
- dbContatos
- dbEscola
- information_schema
- mysql
- test

CodeGen

Dados CSV

CSV para dados MS Excel

Microsoft Excel 2000

Microsoft Word 2000

LaTeX

Abrir Documento Planilha

Abrir Documento de Texto

PDF

SQL

Texy! texto

YAML

Opções

Adicionar comentário pessoal no cabeçalho (\\n quebra linhas)

Comentários

Encapsular exportação numa transação

Desabilitar verificação de chaves estrangeiras

Modo de compatibilidade SQL: NONE

Opções de exportação do Banco de Dados

Adicionar DROP DATABASE

Estrutura

Adicionar DROP TABLE / VIEW / PROCEDURE / FUNCTION / EVENT

Adicionar IF NOT EXISTS

Adicionar valor AUTO_INCREMENT

Usar aspas simples nos nomes de tabelas e campos

Adicionar CREATE PROCEDURE / FUNCTION / EVENT

Adicionar nos comentários

Criar/Atualizar/Verificar datas

Dados

Inserções completas

Inserções estendidas

Tamanho máximo da consulta gerada: 50000

Usar inserções demoradas

Usar inserções ignoradas

Usar hexadecimal para BLOB

Tipo de exportação: INSERT

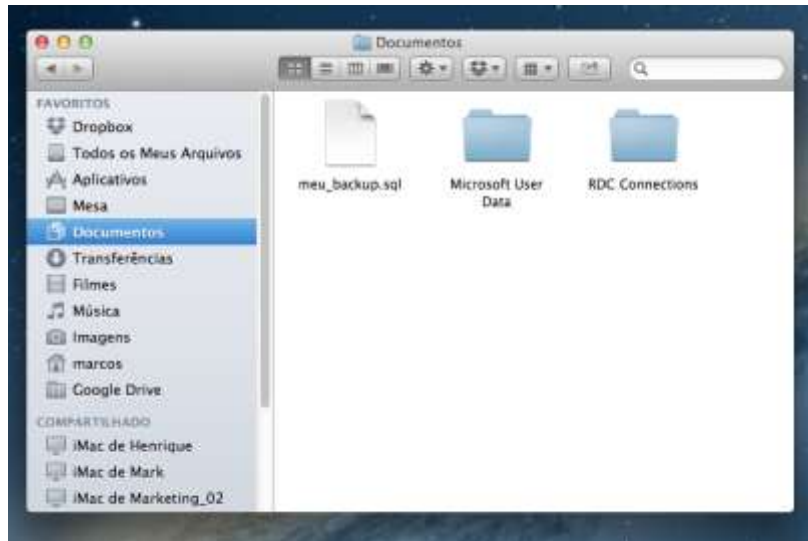
Enviado

Nome do arquivo do modelo¹: Meu_Backup (lembrar modelo)

Compressão: Nenhum "compactado com zip" "compactado com gzip" "compactado com bzip"

Executar

6. Caso o navegador não esteja configurado para salvar arquivos automaticamente, defina a pasta onde deve ser salvo o backup. Por exemplo a pasta **Documentos**.
7. Verifique na pasta onde salvou o arquivo se o mesmo está lá.



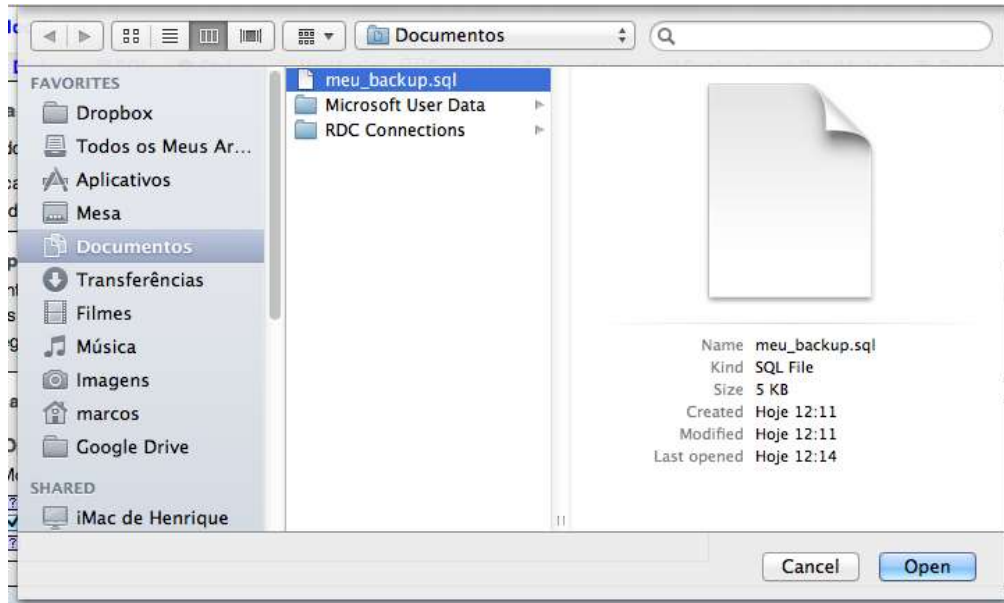
Restaurando um Backup

Um backup de bancos de dados do MySQL é simplesmente uma cópia dos bancos de dados e suas respectivas tabelas e registros gravados em um arquivo de texto em formato de linguagem SQL. Para restaurar os bancos de dados contidos neste arquivo SQL, o phpMyAdmin só precisa executar os comando SQL contidos neste arquivo.

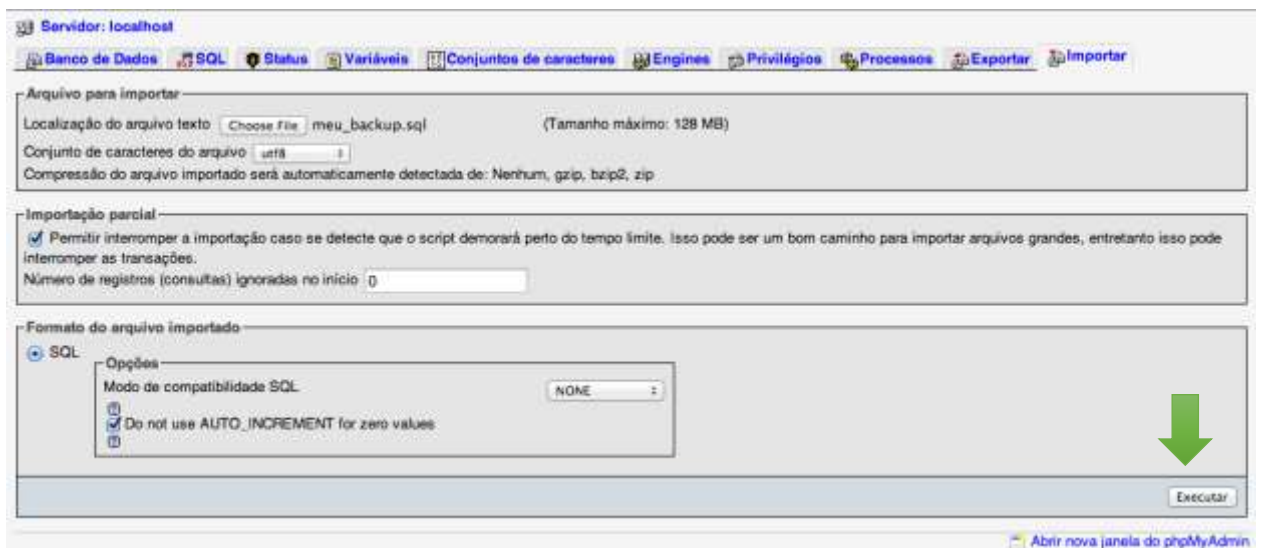
O processo de restauração de um backup de banco de dados pelo phpMyAdmin é bastante simples também. Basicamente é um processo de importação de dados, sem maiores detalhes, deve-se somente encontrar o arquivo de backup e executar a restauração deste backup.

Siga as etapas para restaurar o backup criado anteriormente;

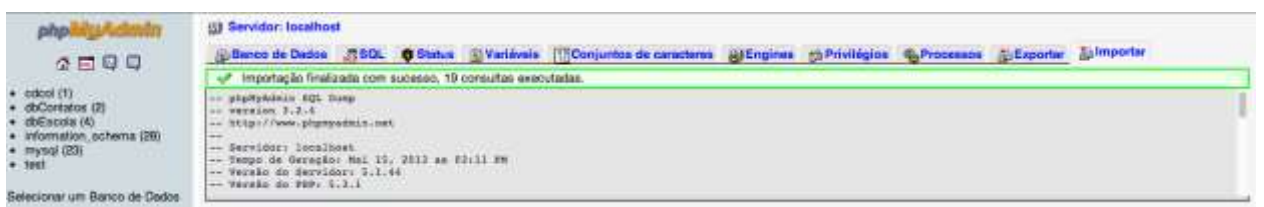
1. Clique na guia **impostar**.
2. Na sessão **Arquivo para importar**, clique no botão **Choose File** para localizar o arquivo de backup.
3. Localize a pasta onde o arquivo está, clique nele e em seguida abra-o.



4. Após indicar o local do arquivo para importar, clique no botão **Executar**.



5. Após executar o backup os bancos de dados são restaurados no sistema.



Aula 11 – Revisão

Esta aula é reservada para a revisão das aulas ministrada no livro. Uma aula com exercícios variados para praticar os comando SQL do MySQL. Estude bastante.

Exercício 1 – Criando um banco de dados e suas tabelas

1. Descreva o comando SQL correto para se criar um banco de dados no servidor MySQL com o nome “dbGeral”. Em seguida teste em seu computador usando o software Workbench.

R:

2. Descreva o comando SQL correto para se criar a tabela de dados com base nas informações a seguir.

Nome da tabela: tbProduto:

Nome do Campo	Tipo de dados	Not Null	Chave Primaria	Auto incrementado
codigoProduto	INT(10)	X	X	X
nomeProduto	VARCHAR(45)	X		
descricaoProduto	LONGTEXT	X		
quantidadeProduto	INT(10)	X		
valorUnitProduto	DECIMAL(10.2)	X		

2. Em uma tabela de banco de dados chamada “tbProdutos”, com aproximadamente 1000 produtos cadastrados, precisamos alterar o valor de todos os produtos definidos no campo “valorProduto” para R\$ 2,50 reais de acréscimo ao valor cadastrado em cada registro. Assinale a alternativa correta que atenda a esta necessidade.

- a) () - UPDATE tbProdutos valorProduto = (valorProduto + 2.5)
- b) () - UPDATE SET tbProdutos valorProduto = valorProduto + 2.5
- c) () - UPDATE tbProdutos valorProduto = (valorProduto + 2.5) WHERE valorProduto = 2.5
- d) () - update tbProdutos SET valorProduto = valorProduto + 2.5
- e) () - UPDATE FROM tbProdutos SET valorProduto = 2.5

3. Considerando ainda uma tabela de banco de dados tbProdutos, com 11 registros cadastrados exibida abaixo, observe os registros existentes na tabela e assinale a alternativa que indica a ação correta a ser executada pela consulta a seguir.

Consulta: `delete from tbProdutos where valorProduto <=2.5 and idProduto >= 8`

Tabela Produtos antes da consulta;

idProduto	valorProduto
1	105.00
2	64.62
3	64.62
4	159.00
5	15.00
6	2.50
7	1.50
8	5.00
9	2.50
10	1.50
11	3.00

- a) () - Foram excluídos 4 registros
- b) () - Foram excluídos todos os registros menores ou iguais a 2.5
- c) () - Foram excluídos 2 registros
- d) () - Foram excluídos todos os registros com idProduto maior ou igual a 8
- e) () - Nem uma das alternativas anteriores

4. Considerando uma tabela de banco de dados chamada tbClientes com 10 registros mostrados a seguir. Assinale a alternativa correta para o resultado retornado da consulta abaixo:

Tabela

idCliente	nomeCliente
1	Johnny Marçal
2	Marcos de Melo
3	Juliana Ap
4	Nicolly Melo
5	Bruno Silva
6	Taina Gomes
7	Sandro Pierobom
8	Lucas Marques
9	Julio Cesar
10	Amanda Nogueira

Consulta:

```
SELECT idCliente, nomeCliente FROM tbClientes WHERE idCliente <4 OR idCliente > 8
```

- a) () - Selecciona somente os registros 1,2,3,9,10
- b) () - Selecciona somente os registros 9,10
- c) () - Selecciona somente os registros 4,5,6,7,8
- d) () - Selecciona somente os registros 1,2,3
- e) () - Selecciona somente os registros 5,6,7

5. Considerando uma tabela de banco de dados chamada tbClientes com 10 registros mostrados a seguir, podemos perceber que, na coluna do campo nomeCliente, os dados estão fora de uma ordem crescente ou decrescente. Assinale a alternativa correta cuja a consulta SQL correta seleciona os dados no campo nomeCliente em ordem crescente ou seja, de A para Z:

Tabela tbClientes antes da consulta;

idCliente	nomeCliente
1	Johnny Marçal
2	Marcos de Melo
3	Juliana Ap
4	Nicolly Melo
5	Bruno Silva
6	Taina Gomes
7	Sandro Pierobom
8	Lucas Marques
9	Julio Cesar
10	Amanda Nogueira

- a) ()- SELECT idCliente, nomeCliente FROM tbClientes order by desc nomeCliente
- b) ()- SELECT idCliente, nomeCliente FROM tbClientes order by nomeCliente asc
- c) ()- SELECT idCliente, nomeCliente FROM tbClientes order by nomeCliente desc
- d) ()- SELECT * FROM tbClientes order by asc nomeCliente
- e) ()- SELECT idCliente, nomeCliente FROM tbClientes order by asc nomeCliente

Aula 12 – Avaliação