MySQL + PHP

- MySQL é um dos sistemas de gerenciamento de banco de dados mais usados da atualidade, que utiliza a linguagem de manipulação de dados chamada SQL (structured query language ou linguagem de consulta estruturada) para acesso aos dados;
- MySQL (adquirido pela Oracle) está disponível sob a licença pública GNU GPL;
- Por ser propriedade de uma empresa que já tem um outro banco de dados comercial, muitos veem o futuro do MySQL com desconfiança. Uma ótima alternativa ao MySQL, verdadeiramente open-source, é o **MariaDB (https://mariadb.org/pt-br/).**

Características do MySQL

- Usado em aplicações onde a velocidade conta mais;
- Não exige grande capacidade de hardware;
- Excelente velocidade de processamento de consultas, mesmo sobre grande base de dados;
- Indicado para situações onde as consultas ao banco de dados não são muito complexas;
- Tem limitações quanto ao tamanho máximo das tabelas armazenadas;
- Compatível com diversos sistemas operacionais.

Interface gráfica para interação com o MySQL

- Existem diversas aplicações destinadas a facilitar a manipulação de dados com o MySQL. Entre elas podemos citar:
  - HeidiSQL - http://www.heidisql.com/
  - PHPMyAdmin - http://www.phpmyadmin.net/home_page/index.php
  - Workbench - http://www.mysql.com/products/workbench

Integração PHP e banco de dados MYSQL

- Daqui por diante, aprenderemos as técnicas envolvendo manipulação do banco de dados MySQL, através de comandos da linguagem PHP.

Regras para nomes no MySQL

- Os nomes utilizados para *base de dados*, *tabelas* e *colunas*:
  - Devem ter até 64 caracteres de comprimento;
  - Podem começar com um número, mas não podem conter exclusivamente números;
  - Caracteres permitidos são: números, letras, o sublinhado _ e o $.

Tipo de dado TEXTO

- **char** – um string de tamanho fixo – permite até 255 caracteres – permite um valor-padrão. Um número máximo de caracteres deve ser especificado no campo;
- **varchar** – um string de tamanho variável. Um número máximo de caracteres deve ser especificado no campo. Armazena até 65535 caracteres – aceita um valor-padrão;
- **text** – armazena até 65535 caracteres. Não aceita um valor-padrão para o campo. Não aceita especificar um tamanho máximo de caracteres para o campo.

Tipo de dado NÚMERICO

- **int** – um número inteiro entre –2.147.483.648 e 2.147.483.647. Se o campo for marcado como não sinalizado, a faixa vai de 0 até 4.294.967.295
- **float** – armazena números decimais – permite definer dois valores para limitar a faixa de abrangência do número: o primeiro refere-se ao número máximo de dígitos do valor e o segundo diz quantos daqueles dígitos deveriam aparecer após a vírgula decimal;
- **decimal** – idem ao tipo de dado float. A única diferença entre os tipos float e decimal é a precisão do valor sendo representado. Float, devido a arredondamentos e precisão aproximada, pode apresentar inconsistências no resultado final calculado. Por isso, para valores reais, decimal é recomendado.

Tipo de dado DATA E HORA

- **date –** armazena uma data no format **YYYY-MM-DD**. A faixa vai de 1000-01-01 até 9999-12-31**;**
- **datetime –** uma combinação de data e hora no formato **YYYY-MM-DD HH:MM:SS;**
- **timestamp –** representação de uma data ou hora convertida em segundos**.**

Tipo de DADO BINÁRIO

- O MySQL permite armazenar, em seus campos, dados no formato binário, como uma imagem, um arquivo de som, etc. . . , embora isso não seja adequado. Os tipos de dados que armazenam campos binários são:
  - **tinyblob** – dados binários até 255 bytes;
  - **blob** – dados binários até 64 KB;
  - **mediumblob** – até 16 MB;
  - **longblob** – até 4 GB.
  - **blob** = binary long object.

Tipo de dado BOOLEANO (bool ou boolean)

- Nativamente, o MySQL não apresenta um tipo de dado específico para a representação de valores lógicos utilizando-se ***true*** ou ***false***. Em vez disso, usamos bool ou boolean que, internamente, faz o MySQL converter o dado para **tinyint(1)**. Um valor de 0 para tinyint é considerado falso. Qualquer outro valor é considerado verdadeiro. Assim, podemos fazer:

```
CREATE TABLE conta_bancaria(tem_conta boolean);

INSERT INTO conta_bancaria(false);

INSERT INTO conta_bancaria(0);

INSERT INTO conta_bancaria(true);

INSERT INTO conta_bancaria(1);

INSERT INTO conta_bancaria('SIM');
```

Passos para a interação entre MySQL e PHP

1. Fazer a conexão com o servidor MySQL*;
2. Criar o banco de dados, se ele ainda não existir;
3. Selecionar o banco de dados*;
4. Criar a tabela, se ainda não existir;
5. Realizar operações sobre a tabela: inserção, exclusão, alteração, etc...*;
6. Fechar a conexão com o banco de dados*.

* = passos obrigatórios.

Passo 1 – Abrir a conexão com o banco de dados

- Abrir uma conexão é o primeiro passo antes de qualquer outra atividade com o banco de dados;
- Usamos o construtor **mysqli(servidor, usuário, senha);** Exemplo:

```
$conexao = new mysqli("localhost", "root", "aluno");
```

O comando acima cria o objeto $conexão, a partir da classe **MySQLi**. Esta classe contém numerosos métodos e propriedades para permitir a comunicação entre o PHP e o MySQL.

Veja a descrição completa desta classe em:
https://secure.php.net/manual/pt_BR/class.mysqli.php

Passo 2 – Criação da base de dados (opcional)

- Para criarmos, em PHP, uma base de dados de nome biblioteca, por exemplo, faríamos como se segue:

```
$sql = "CREATE DATABASE IF NOT EXISTS biblioteca";

$resultado = $conexao->query($sql) or die($conexao->error);
```

- Para executarmos qualquer query, utilizamos o método query do objeto conexão criado anteriormente. Se houver algum erro na operação anterior, podemos verifica-lo por meio do uso da propriedade error, e encerrar o script.
- A execução de uma query em PHP sempre produz um objeto do tipo resultado. Este objeto é implementado por meio da classe **MySQLi_Result**. Esta classe contém diversos métodos e propriedades que o PHP utiliza para manipular os dados recebidos do banco de dados. Veja mais em: http://php.net/manual/pt_BR/class.mysqli-result.php

Passo 3 - Seleção da base de dados

- Após feita a conexão, o PHP exige que você especifique qual o banco de dados que estará em uso;
- Exemplo: o comando abaixo abre o banco de dados "biblioteca" e o deixa pronto para uso:

```
$conexao->select_db("biblioteca");
```

- Pode-se usar variáveis do PHP no lugar do nome da base de dados.

Passo 4 – Criação da tabela (opcional)

- Para criarmos, automaticamente, por exemplo, uma tabela chamada alunos com os campos nome e média, faríamos:

```
$sql = "CREATE TABLE IF NOT EXISTS alunos(nome varchar(30), media decimal(3,2));
$resultado = $conexao->query($sql) or die($conexao->error);
```

- A consulta define os nomes dos campos, tipo de dados e outros atributos.
- A execução da query gera um objeto resultado, ou uma mensagem de erro com a interrupção do script em PHP, se for o caso.

Passo 5 – operações sobre a base de dados

- Nesta etapa, devemos implementar, em PHP, os diversos comandos que realizam as mais variadas operações sobre os dados da base de dados, tais como:
  - Exclusão de tabelas, base de dados ou registros;
  - Inserção de novos registros na tabela;
  - Alteração dos dados de uma tabela;
  - Listagem das informações de determinados campos da tabela na página web;
  - Etc...

Passo 6 - Fechamento da conexão

- Ao usarmos bancos de dados que se baseiam na linguagem de manipulação SQL, é imprescindível que, após o término das operações, o banco seja fechado;
- Para isto, basta, em PHP, encerrarmos a conexão;
- Isso é feito através do método close() do objeto que armazena a conexão com o banco de dados: **$conexao->close();**

Comandos da linguagem SQL no PHP

- Depois de criada uma conexão e selecionada a base, podemos utilizar todos os comandos da linguagem SQL para manipularmos o banco de dados;
- Toda consulta (query, dentro da variável $sql) à base de dados é feita pelo método:

```
$resultado = $conexao->query($sql);
```

Métodos importantes das classes MySQLi e MySQLi_Result do PHP

Método/propriedade	Descrição
affected_rows	Esta propriedade retorna o número de registros afetados pela consulta imediatamente anterior
mysqli_error	Esta propriedade retorna para o PHP a mensagem de erro do MySQL ocasionada por falha em determinada consulta
query()	Método que envia para o banco de dados determinada consulta
fetch_array()	Método que recupera um registro de uma tabela. Deve estar dentro de um laço. O registro é armazenado em um vetor, que pode ser acessado com índice numérico ou associativo. O índice associativo é o próprio nome da coluna da tabela