Programação II

Banco de Dados

Banco de Dados

- Segundo Korth, um banco de dados “é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico”.
  - sempre que for possível agrupar informações (em meio eletrônico ou não) que se relacionam e tratam de um mesmo assunto, temos um banco de dados.
- Já um sistema de gerenciamento de banco de dados (SGBD) é um software que possui recursos capazes de manipular as informações do banco de dados e interagir com o usuário.
  - Exemplos de SGBDs: Oracle, SQL Server, DB2, PostgreSQL, MySQL, MariaDB, Microsoft Access, entre outros.

Sistema Gerenciador de Banco de dados

Banco de Dados Relacional

- Um BD relacional representa os dados como conjuntos de tabelas, onde as colunas são os atributos (ou campos) e as linhas são os registros (ou tuplas).
  - Cada tabela possui uma chave primária (que pode ser formada por uma ou mais colunas).
  - Relacionamentos entre as tabelas são estabelecidos por meio de chaves estrangeiras (colunas que apontam para as chaves primárias de outras tabelas).

Banco de Dados Relacional

PRODUTO

cod_prod	nome	qualidade
P01	laranja	1a
P02	laranja	2a
P03	soja	1a
P04	arroz	1a
P05	arroz	2a
P06	cacau	1a
P07	trigo	2a
P08	pêssego	1a
P09	pêssego	2a
P10	uva	1a
P11	uva	2a

FORNECEDOR

cod_forn	nome	cidade	estado
F01	Pedro	Porto Alegre	RS
F02	Eliana	Botucatu	SP
F03	Olacyr	Curitiba	PR
F04	João	Pelotas	RS
F05	Ernesto	Anápolis	GO
F06	Mário	Limeira	SP
F07	Hans	Bento Gonçalves	RS
F09	Antônio	Anápolis	GO
F10	Mário	Curitiba	PR

ESTOQUE

cod_forn	cod_prod	qtde	procedência
F01	P01	100	Araraquara
F01	P02	150	Limeira
F01	P10	200	Bento Gonçalves
F01	P11	130	Vinhedo
F02	P07	240	Maringá
F02	P08	260	Pelotas
F02	P09	190	Bento Gonçalves
F03	P03	320	Maringá
F03	P07	210	Maringá
F03	P06	200	Ilhéus
F05	P04	150	Catalão
F05	P05	270	Uberlândia
F06	P01	80	Bebedouro

PEDIDO

cod_ped	cod_forn	cod_prod	qtde	loc_armaz
1001	F06	P02	120	Limeira
1002	F07	P10	110	Bento Gonçalves
1003	F07	P11	130	Pelotas
1004	F09	P04	100	Catalão
1005	F09	P07	80	Maringá
1006	F10	P03	220	Maringá

MySQL / MariaDB

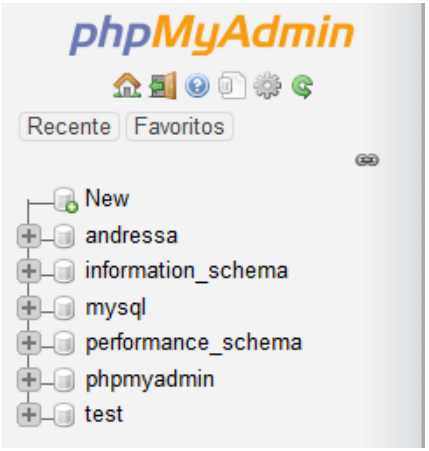
- O MySQL é um SGBDR (Sistema Gerenciador de Banco de Dados Relacional) gratuito e *Open Source,* desenvolvido para aplicações Web, onde ocorrem em geral mais consultas do que atualizações.
- Utiliza o modelo cliente / servidor:
  - Há um *daemon* rodando permanentemente num servidor;
  - Os clientes acessam o banco via conexões TCP/IP, informando o IP do servidor e a porta do serviço (a porta padrão no Windows/Linux é 3306).
- Não vem com interface gráfica. Pode-se administrá-lo totalmente através de comandos SQL (pelo utilitário mysql monitor, que acompanha o servidor), ou instalando uma interface gráfica em separado:
  - Ex: phpMyAdmin (acompanha o XAMPP), MySQL Workbench, MySQL Front
- Após a aquisição do MySQL pela Oracle, seus desenvolvedores criaram um *fork* do projeto chamado **MariaDB**, o qual funciona exatamente como o MySQL.
  - Todos os comandos, interfaces, bibliotecas e APIs que existem no MySQL também existem no MariaDB. Não há necessidade de adaptações.

Instalação do MySQL

- Se você instalou o XAMPP, o MySQL (na verdade o MariaDB) faz parte do pacote. Basta então iniciar o serviço pelo painel do XAMPP.
- O phpMyAdmin pode então ser acessado pelo endereço:

  http://localhost/phpmyadmin

  - Se você não cadastrou um usuário, e está usando a instalação do Xampp, basta logar com o usuário e a senha root.
  - Caso contrário, utilize sua senha e usuário do banco de dados.



- Para acessar pelo terminal, acesse a pasta /bin da instalação do XAMPP (/opt/lampp/bin ou c:\xampp\mysql\bin) e digite:
  - Windows: `mysql -u root -p`
  - Linux: `./mysql -u root -p`

MySQL: Principais Tipos de Dados

- **INT**: inteiros entre -2.147.483.648 e 2.147.483.647
- **SMALLINT**: inteiros de -32768 a 32767
- **MEDIUMINT**: inteiros de -8388608 a 8388607
- **BIGINT**: inteiros entre -9.223.372.036.854.775.808 e 9.223.372.036.854.775.807
- **DECIMAL(M,D)**: ponto decimal com M dígitos no total (precisão) e D casas decimais (escala); o padrão é 10,0; M vai até 65 e D até 30.
- **FLOAT(M,D)**: ponto flutuante com precisão M e escala D; o padrão é 10,2; D vai até 24.
- **CHAR(M)**: string que ocupa tamanho fixo entre 0 e 255 caracteres
- **VARCHAR(M)**: string de tamanho variável (até 65535 caracteres).
- **BOOL / BOOLEAN**: valores binários 0 / 1; Na verdade, é um alias para o tipo TINYINT(1)
- **BLOB**: campo com tamanho máximo de 65535 caracteres binários; Binary Large Objects, são usados para armazenar grandes quantidades de dados, como imagens.
- **MEDIUMTEXT**: até 16.777.215 caracteres.
- **LONGTEXT**: até 4.294.967.295 caracteres.
- **DATE**: data de 01/01/1000 a 31/12/9999, no formato YYYY-MM-DD
- **DATETIME**: combinação de data e hora de 01/01/1000 00:00:00 a 31/12/9999 23:59:59, no formato YYYY-MM-DD HH:MM:SS
- **TIME**: hora apenas, no formato HH:MM:SS
- **YEAR(M)**: ano nos formatos de 2 ou 4 dígitos;

Linguagem SQL

- **SQL**: *Structured Query Language* (Linguagem de Consulta Estruturada).
- Linguagem padrão para a manipulação de bancos de dados relacionais.
- Subconjunto DDL – *Data Definition Language*:
  - **Create** (criar estruturas como database, table, user, view, procedure, etc)
  - **Alter** (alterar estruturas)
  - **Drop** (apagar estruturas)
  - **Grant** (conceder privilégios de acesso)
- Subconjunto DML – *Data Manipulation Language*:
  - **Insert** (inserir registros)
  - **Select** (consultar registros)
  - **Update** (alterar registros)
  - **Delete** (apagar registros)

Estas 4 operações são conhecidas como CRUD: Create, Read, Update e Delete

Exemplo – Criar database e tabela

- Criando um database:
  - CREATE DATABASE **Site** DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
- Selecionando um database (no phpMyAdmin, basta clicar no database):
  - use **Site**;
- Criando uma tabela:
  - CREATE TABLE **cliente** (

```
id INTEGER AUTO_INCREMENT PRIMARY KEY,
nome VARCHAR(20) NOT NULL,
login VARCHAR(20) NOT NULL,
senha VARCHAR(20) NOT NULL,
email VARCHAR(50) NOT NULL,
dataNascimento DATE,
categoria CHAR,
ativo BOOLEAN
);
```

Para modificar: ALTER TABLE...
Para excluir: DROP TABLE...

Exemplo – Permissões

- Criando usuário:
  - CREATE USER '**master**'@'localhost' IDENTIFIED BY '**senha**';

Pode ser necessário criar para 127.0.0.1

- Concedendo privilégios:
  - GRANT SELECT, INSERT, UPDATE, DELETE ON **Site.*** TO '**master**'@'localhost';
  - GRANT ALL ON …

Todas as tabelas do database Site.
Pode-se apontar tabelas específicas.

- Revogando privilégios:
  - REVOKE DELETE ON **Site.*** FROM '**master**'@'localhost';

- Excluindo usuário:
  - DROP USER '**master**'@'localhost';

Exemplo – Insert

- Inserindo valores para todos os campos (exceto id, que é AUTO_INCREMENT):
  - INSERT INTO cliente (nome, login, senha, email, dataNascimento, categoria, ativo) VALUES ('João da Silva', 'jsilva', '12345', 'jsilva@gmail.com', '1990-01-20', '1', 1);

- Inserindo somente campos obrigatórios (NOT NULL):
  - INSERT INTO cliente (nome, login, senha, email) VALUES ('João da Silva', 'jsilva@gmail.com', 'jsilva', '12345');

- Inserindo mais de um registro simultaneamente:
  - INSERT INTO cliente (nome, login, senha, email) VALUES ('João da Silva', 'jsilva@gmail.com', 'jsilva', '12345'), ('Pedro dos Santos', 'psantos@gmail.com', 'psantos', '12345'), ('Luiza Mota', 'lmota@gmail.com', 'lmota', '12345');

Exemplo – Select

- Selecionando todos os campos:
  - SELECT * FROM cliente;
- Selecionando apenas alguns campos:
  - SELECT nome, login FROM cliente;
- Selecionando e aplicando filtros:
  - SELECT nome, login FROM cliente WHERE ativo = 0;
  - SELECT nome, login, email FROM cliente WHERE email LIKE '%@gmail.com%';
  - SELECT nome, login, email FROM cliente WHERE email LIKE '%@gmail.com%' and categoria = 3;

Exemplo – Update

- Atualizar o e-mail do cliente:
  - update cliente SET email = 'joaozinho@hotmail.com' WHERE id = 1;
- Alterando vários campos:
  - update cliente SET nome = 'João da Silva Sauro', login = 'jsauro', senha='65432', email = 'joaozinho@hotmail.com' WHERE id = 4;
- Se a cláusula WHERE for omitida, todos os registros serão alterados (cuidado!!!)

Exemplo – Delete

- Apagando um registro:
  - delete from cliente where id = 4;

- Apagando todos os registros:
  - delete from cliente;

Backup da base via terminal

- Considerando que o MySQL está rodando no XAMPP, localize o arquivo mysqldump (/opt/lampp/bin ou c:\xampp\mysql\bin):
  - para exportar todas as bases de dados para o arquivo backup.sql:
    - $ ./mysqldump -u root -p --all-databases > backup.sql
  - para importar uma das bases de dados do arquivo backup.sql:
    - $ ./mysql -u root -p --one-database nome-da-base < backup.sql
  - para exportar apenas uma base de dados:
    - $ ./mysqldump -u root -p nome-da-base > backup.sql
  - para importar uma base de dados:
    - $ ./mysqldump -u root -p nome-da-base < backup.sql

Acessando o MySQL através do PHP

- Há 2 formas:
  - Utilizando a extensão mysqli (substitui a antiga extensão mysql).
  - Utilizando a extensão PDO - *PHP Data Objects*, uma camada de abstração que utiliza os mesmos comandos para acessar dados de diversos SGBD.

- Por questões de simplicidade, utilizaremos a primeira. São necessários 4 passos:

  1) Estabelecer uma conexão (**`mysqli_connect`**);

  2) Enviar um comando SQL para ser executado (**`mysqli_query`**);

  3) Tratar o retorno
     - Se a query for um select, será retornado um conjunto de resultados;
     - Se for outra query qualquer, retornará verdadeiro (sucesso) ou falso (erro).

  4) Encerrar a conexão (**`mysqli_close`**).

mysqli_connect()

- Abre uma conexão com o servidor MySQL especificado. A sintaxe da função é mostrada abaixo:

int mysqli_connect(string [host[:porta]] , string [login] , string [senha],string [database]);

- A função retorna um valor inteiro: zero, em caso de erro, ou o identificador da conexão, em caso de sucesso.
- Para que possamos referenciar posteriormente o identificador da conexão estabelecida, devemos armazená-lo numa variável, conforme o exemplo abaixo:

```
$conexao = mysqli_connect("localhost", "master", "12345", "Site");
```

Dica:

- Podemos colocar os comandos que criam a conexão dentro de um arquivo separado, e inclui-lo nas demais páginas através do comando include, include_once, require ou require_once. Assim, caso o usuário, senha, servidor ou nome do database forem alterados, não precisaremos alterar todas as páginas, mas sim apenas o arquivo que cria a conexão.

conexao.php

```
<?php
  $host = "localhost";
  $user = "master";
  $senha = "12345";
  $database = "Site";
  $conexao = mysqli_connect($host, $user, $senha, $database) or
   die("Houve um erro de conexão ao banco de dados.");
?>
```

Charset

- Se tiver problemas com a codificação, acrescente as seguintes linhas ao arquivo de conexão:

```
mysqli_query($conexao, "SET NAMES 'utf8'");
mysqli_query($conexao, 'SET character_set_connection=utf8');
mysqli_query($conexao, 'SET character_set_client=utf8');
mysqli_query($conexao, 'SET character_set_results=utf8');
```

mysqli_query()

- Este comando é utilizado para enviar declarações SQL ao servidor. A sintaxe é:

int mysqli_query(int conexao, string query);

- Query é a declaração SQL, e conexão é o identificador da conexão, como no exemplo:

```
$resultado = mysqli_query($conexao, "select * from cliente");
```

- A expressão SQL será executada no banco de dados selecionado anteriormente pelo comando `mysqli_connect()`;
- Armazenamos o valor de retorno numa variável para poder manipulá-lo adequadamente. A função retorna falso (zero) se a expressão SQL for incorreta, e diferente de zero se for correta.

Tratando um conjunto de resultados

- Quando a declaração SQL enviada ao banco for uma expressão SELECT, as linhas retornadas pela consulta são armazenadas numa memória de resultados, e a função retorna um valor interno que identifica o resultado (Resource ID).
- Podemos manipular os dados retornados de várias formas:
  - `mysqli_free_result($resultado)`: apaga da memória o resultado indicado;
  - `mysqli_num_rows($resultado)`: retorna o número de linhas contidas num resultado;
  - `mysqli_fetch_array($resultado)`: lê uma linha do resultado e devolve um vetor, cujos índices são numéricos. A execução repetida do mesmo comando lê a próxima linha, até chegar ao final do resultado.
  - `mysqli_fetch_assoc($resultado)`: lê uma linha do resultado e devolve um array associativo, cujos índices são os nomes das colunas no BD. A execução repetida do mesmo comando lê a próxima linha, até chegar ao final do resultado.

mysqli_close()

- O PHP automaticamente encerra todas as conexões com bancos de dados ao final da execução do script que abriu a conexão. Para encerrar explicitamente uma conexão, utilize:

int mysqli_close(int identificador da conexão);

- Ex:

```
mysqli_close($conexao);
```

mysqli_error()

- Permite acessar as mensagens de erro retornadas pelo banco de dados MySQL. O parâmetro é o identificador da conexão:

```
echo mysqli_error($conexao);
```

- O comando acima retorna sempre o último erro ocorrido na conexão identificada pela variável `$conexao`.

Exemplo de Seleção

```
<h1>Cadastro de Clientes</h1>
  <table border="1">
    <tr>
      <th>NOME</th><th>LOGIN</th><th>E-MAIL</th><th>DATA DE
NASCIMENTO</th><th>CATEGORIA</th><th>ATIVO</th>
    </tr>
    <?php
      require_once "conexao.php";
      $resultado = mysqli_query($conexao, "select nome, login, email,
DATE_FORMAT(dataNascimento, '%d/%m/%Y') as data, categoria, ativo from cliente order by
nome");
      if (mysqli_num_rows($resultado) == 0)  {
          echo '<tr><td colspan="6">Nenhum cliente cadastrado.</td></tr>';
      }
      else {
        while ($row = mysqli_fetch_array($resultado)) {
           echo "<tr><td>".$row['nome']."</td><td>".$row['login']."</td><td>".
$row['email']."</td><td>".$row['data']."</td><td>".$row['categoria']."</td><td>".
$row['ativo']."</td></tr>";
        }
      }
      mysqli_close($conexao);
    ?>
  </table>
```

Exemplo de Inserção

```
<?php
  require_once "conexao.php";
  $nome = $_POST['nome'];
  $login = $_POST['login'];
  $senha = $_POST['senha'];
  $email = $_POST['email'];
  $dataNascimento = $_POST['dataNascimento'];
  $categoria = $_POST['categoria'];
  $ativo = $_POST['ativo'] == 1 ? 1 : 0;
  $resultado = mysqli_query($conexao, "insert into cliente (nome, login, senha,
   email, dataNascimento, categoria, ativo) values ('$nome', '$login',
   '$senha', '$email', '$dataNascimento', '$categoria', $ativo);");
  if ($resultado)  {
        echo "Cliente cadastrado com sucesso.";
  }
  else  {
    echo "Erro. O usuário não pôde ser cadastrado.";
    echo mysqli_error($conexao);
  }
  mysqli_close($conexao);
?>
```

Não é recomendável concatenar as entradas do usuário diretamente na instrução SQL.
Aqui também devemos acrescentar outros comandos de tratamento e validação dos dados.

Este exemplo pressupõe que há um formulário com método POST e todos os campos citados

Alterando dados

- Para alterar o conteúdo de um registro, precisamos primeiramente que o usuário indique qual o registro a ser editado, para então carregarmos as informações atuais a partir do BD e disponibilizarmos as mesmas num formulário de edição.
- Uma estratégia é colocar um link “editar” ao lado de cada registro Este link aponta para a página que carrega as informações do BD, enviando como parâmetro na URL alguma informação que identifique unicamente o registro (normalmente o valor da chave primária). Ex:

```
<a href="update.php?id=2">editar</a>
```

- Assim, no exemplo acima, a página `update.php` terá uma variável **$_GET['id']** com o valor 2, o qual será utilizado como filtro de busca.

Exemplo de Formulário de Edição

```
<h1>Alteração de Clientes</h1>
<?php
  require_once "conexao.php";
  $id = $_GET['id'];
  $resultado=mysqli_query($conexao, "select * from cliente where id=$id");
  $cliente = mysqli_fetch_array($resultado);
?>
<form method="post">
  <input name="id" type="hidden" value="<?=$cliente['id'];?>">
  <label>Nome:
  <input type="text" name="nome" value="<?=$cliente['nome'];?>"></label><br><br>
  <label>Login:
  <input type="text" name="login" value="<?=$cliente['login'];?>"></label><br><br>
  <label>Senha:
  <input type="password" name="senha" value="<?=$cliente['senha'];?>"></label><br><br>
  <label>E-mail:
  <input type="text" name="email" value="<?=$cliente['email'];?>"></label><br><br>
  <label>Data de Nascimento:
  <input type="text" name="dataNascimento" value="<?=$cliente['dataNascimento'];?>"></label><br><br>
  <label>Categoria:
    <select name="categoria">
      <option value="1" <?php echo ($cliente['categoria'] == 1) ? "selected": "";?>>Ouro</option>
      <option value="2" <?php echo ($cliente['categoria'] == 2) ? "selected": "";?>>Prata</option>
      <option value="3" <?php echo ($cliente['categoria'] == 3) ? "selected": "";?>>Bronze</option>
    </select>
  </label><br><br>
  <label><input type="checkbox" name="ativo" value="1" <?php echo ($cliente['ativo'] == 1) ?
   "checked": "";?>>Ativo</label><br><br>
  <input type="submit" value="Alterar dados" name="alterar"> <a href="index.php">Cancelar</a>
</form>
```

Exemplo de Edição

```
<?php
  if (isset($_POST['alterar'])){
    require_once "conexao.php";
    $id = $_POST['id'];
    $nome = $_POST['nome'];
    $login = $_POST['login'];
    $senha = $_POST['senha'];
    $email = $_POST['email'];
    $dataNascimento = $_POST['dataNascimento'];
    $categoria = $_POST['categoria'];
    $ativo = $_POST['ativo'] == 1 ? 1 : 0;
    $resultado = mysqli_query($conexao, "update cliente set nome = '$nome', login =
   '$login', senha = '$senha', email = '$email', dataNascimento = '$dataNascimento',
    categoria = '$categoria', ativo = $ativo where id=$id;");
    if ($resultado)  {
      echo "Cliente alterado com sucesso.";
    }
    else  {
      echo "Erro. O usuário não pôde ser alterado.";
      echo mysqli_error($conexao);
    }
    mysqli_close($conexao);
  }
?>
```

Não é recomendável concatenar as entradas do usuário diretamente na instrução SQL.
Aqui também devemos acrescentar outros comandos de tratamento e validação dos dados.

Campos HIDDEN

- Note que no formulário de edição utilizamos um campo do tipo **HIDDEN** (oculto). Este campo serve para armazenar o id do cliente que foi passado pelo link “editar”.
- Como o campo HIDDEN está dentro do formulário, quando o mesmo for submetido via POST, a página terá a variável $_POST['id'], contendo o código do cliente a ser alterado. Este dado é utilizado na cláusula WHERE da declaração UPDATE enviada ao BD.
- Os campos HIDDEN servem para passar dados de um formulário para o script especificado no action (ou para uma nova exibição da própria página, se o action não estiver presente) sem que estes dados sejam visíveis e editáveis, como é o caso dos demais campos visíveis (input text, select ou textarea).

Excluindo dados

- Da mesma forma que na alteração, para excluirmos dados da tabela cliente, precisamos que o usuário indique a notícia a ser excluída. Podemos fazer isso da mesma forma que fizemos na alteração, colocando um link “excluir” ao lado de cada notícia, o qual passa como parâmetro o código.

```
<a href="delete.php?id=2">excluir</a>
```

- O ideal se exiba uma confirmação antes da exclusão propriamente dita. Para o exemplo a seguir, suponha que temos um formulário de confirmação com um botão de nome ‘delete’.

Exemplo de Exclusão

```
<?php
  if (isset($_POST['delete'])){
    require_once "conexao.php";
    $resultado = mysqli_query($conexao, "delete from
cliente where id=$_POST[id];");
    if ($resultado)  {
      echo "Cliente excluído com sucesso.<br>";
    }
    else  {
      echo "Erro. O usuário não pôde ser excluído.";
      echo mysqli_error($conexao);
    }
    mysqli_close($conexao);
  }
?>
```