Integração de PHP e MySQL

Apresentação

A linguagem PHP dispõe de ferramentas para acesso a dados que permitem a conexão com sistemas de gerenciamento de banco de dados (SGBD), como, por exemplo, o MySQL. Essas ferramentas, além de fazer a conexão com o banco, o armazenamento e a recuperação dos dados, têm funções de inserção, atualização, exclusão e consulta aos dados.

Nesta Unidade de Aprendizagem, você vai saber como se dá a utilização do PHP em conjunto com um SGBD, mais especificamente com o MySQL. Serão abordados os procedimentos necessários para permitir o gerenciamento e a consulta das informações armazenadas.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Descrever os procedimentos necessários para a integração e a conexão do PHP com o MySQL.
- Identificar os comandos para a manipulação das informações armazenadas nas bases de dados.
- Aplicar comandos para consultas sobre bases de dados MySQL.

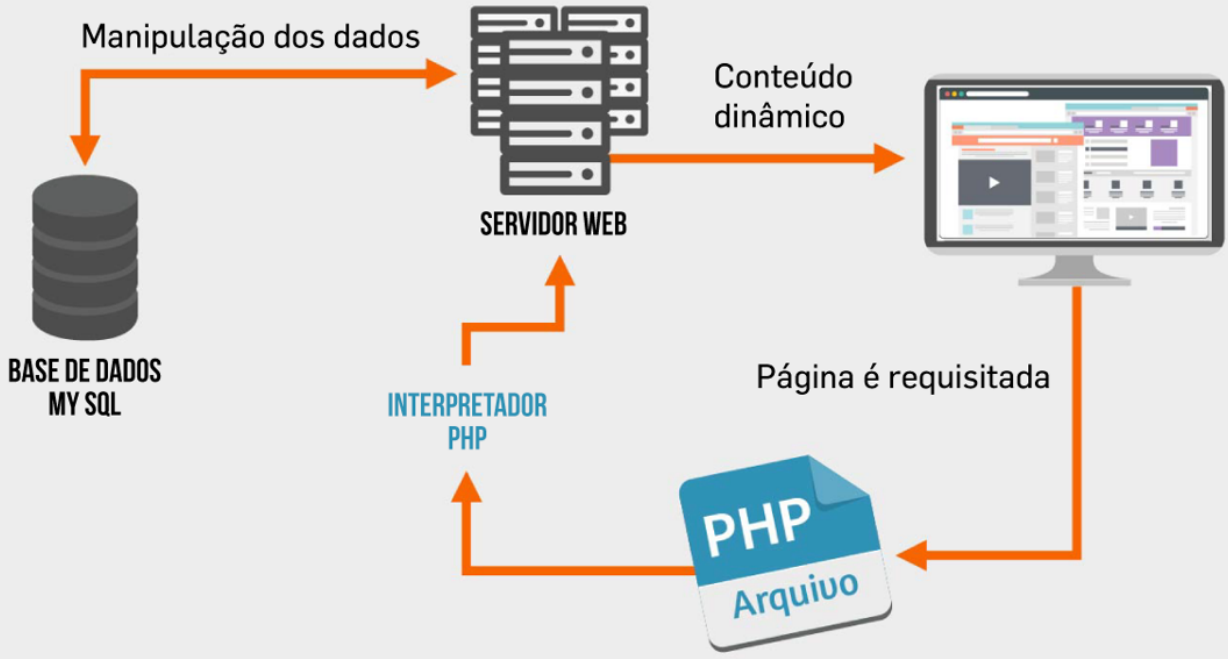
Infográfico

MySQL é um SGBD *open source*, mundialmente conhecido, que permite a criação de banco de dados com suas tabelas de maneira rápida e com custo baixo. Isso faz com que se tenha produtos de *performance* relevante, podendo ser escalável, ou seja, permitindo o crescimento de suas aplicações, desenvolvidas por exemplo em PHP, de modo que sua atualização seja planejada para crescer.

Neste Infográfico, você vai saber como é o processo de obtenção das informações no PHP em conjunto com o MySQL.

PROCESSO DE OBTENÇÃO DE INFORMAÇÕES NO PHP E MYSQL

As informações são requisitadas pelo navegador *web* (*browser*) via arquivo PHP, que, ao passar pelo interpretador, faz o relacionamento dos dados com o SGBD MySQL, retornando os dados ao Servidor Web, que, por sua vez, retorna o conteúdo dinâmico ao navegador.



Já o SGBD MySQL funciona do lado do Servidor Web e tem como funcionalidade o armazenamento de informações dentro de tabelas criadas especialmente para o funcionamento de aplicações web que precisam manter os dados seguros e que permitem, também, a recuperação desses dados.

Como foi possível observar na ilustração, todas as ferramentas (PHP, MySQL e Servidor Web) devem trabalhar em conjunto para que o desenvolvimento de uma aplicação *web*, em PHP, tenha sucesso. Saber como tudo isso funciona é parte fundamental desse sucesso.

Conteúdo do Livro

O PHP e o MySQL são duas das tecnologias mais importantes para o desenvolvimento de sistemas *web*. O PHP é uma linguagem de programação de código aberto extremamente popular e utilizada por diversos desenvolvedores ao redor do mundo. Já o MySQL é um sistema gerenciador de banco de dados relacional, também de código aberto, que oferece alta *performance* e escalabilidade para aplicações *web*.

Juntos, o PHP e o MySQL possibilitam a criação de sistemas dinâmicos, que interagem com o usuário em tempo real e permitem a manipulação de dados de forma eficiente e segura.

Neste capítulo **Integração de PHP e MySQL**, base teórica desta Unidade de Aprendizagem, você vai ver como integrar o PHP com o MySQL. Além disso, vai conhecer comandos para manipular informações armazenadas nas bases de dados e realizar consultas de forma fácil e rápida.

Boa leitura.

FERRAMENTAS DE DESENVOLVIMENTO *WEB*

sagah

Integração de PHP e MySQL

Marcelo da Silva dos Santos

OBJETIVOS DE APRENDIZAGEM

- Descrever os procedimentos necessários para a integração e a conexão do PHP com o MySQL.
- Identificar os comandos para a manipulação das informações armazenadas nas bases de dados.
- Aplicar comandos para consultas sobre bases de dados MySQL.

Introdução

O PHP e o MySQL são duas tecnologias fundamentais para o desenvolvimento *web*. O PHP é uma linguagem de programação do lado do servidor, utilizada para criar aplicações *web* dinâmicas, como sistemas de gerenciamento de conteúdo e lojas virtuais. Já o MySQL é um sistema de gerenciamento de banco de dados relacional, responsável por armazenar e gerenciar os dados de uma aplicação.

A combinação dessas tecnologias permite a criação de *sites* dinâmicos e interativos, capazes de processar informações e exibi-las ao usuário de forma personalizada. Além disso, a utilização do PHP e do MySQL é muito comum em projetos de diferentes tamanhos e complexidades, desde pequenos *sites* até grandes sistemas corporativos.

Neste capítulo, você vai ver como integrar o PHP com o MySQL e fazer diversas operações com banco de dados. Além disso, vai estudar os procedimentos necessários para conectar o PHP com o MySQL, identificar os principais comandos para a manipulação de informações e realizar consultas sobre bases de dados

MySQL. Por fim, vai ver como utilizar o PHP e o MySQL para criar aplicações *web* dinâmicas e interativas.

Utilização de MySQL em aplicações PHP

Os sistemas para internet tornaram-se cada vez mais importantes em nosso mundo altamente conectado e digital, permitindo a criação de aplicações *web* dinâmicas e interativas que oferecem soluções para diversos problemas e necessidades em várias áreas. Entre as linguagens de programação para desenvolvimento de sistemas para internet, o PHP se destaca como uma das mais populares e amplamente utilizadas, em razão de sua facilidade de uso, flexibilidade, grande comunidade de desenvolvedores e capacidade de se integrar com diversos sistemas de gerenciamento de bancos de dados (BASSO, 2014).

Além disso, é importante ressaltar que, para o desenvolvimento de sistemas para internet, o armazenamento de dados é fundamental para permitir que as informações sejam acessadas e gerenciadas de forma rápida e eficiente. Nesse sentido, os sistemas de gerenciamento de banco de dados desempenham um papel crucial no armazenamento, na organização e na recuperação de dados para as aplicações *web*, permitindo que as informações sejam gerenciadas com segurança e escalabilidade.

O banco de dados é uma parte fundamental de muitas aplicações PHP, pois é onde a aplicação armazena e gerencia os dados essenciais para o funcionamento da aplicação. O banco de dados é usado para armazenar informações como nome de usuário, senhas, informações de pedidos em uma loja virtual, conteúdo em um *blog*, etc. Sem um banco de dados adequado, a aplicação não poderia armazenar e gerenciar os dados dos usuários, o que limitaria severamente sua funcionalidade. Além disso, o banco de dados permite que a aplicação seja escalável, uma vez que ela pode armazenar um grande número de dados e gerenciá-los eficientemente.

Entre os diversos sistemas de gerenciamento de banco de dados disponíveis, o MySQL se destaca como uma opção popular e confiável, amplamente utilizada em todo o mundo para o desenvolvimento de aplicações *web* de alta qualidade (BASSO, 2014). Essa popularidade existe porque tanto o PHP quanto o MySQL são de código aberto, são compatíveis com várias plataformas e têm uma grande comunidade de desenvolvedores que contribuem para o seu desenvolvimento e suporte.

Além disso, o MySQL é compatível com a linguagem SQL, que é usada para realizar operações em bancos de dados, como inserir, atualizar e recuperar dados. O PHP também oferece várias funções e extensões que permitem que os desenvolvedores se conectem facilmente ao MySQL e realizem operações em bancos de dados a partir do código PHP.

Saiba mais

"O MySQL foi disponibilizado, em sua primeira versão, em 1996, apesar de já ter ocorrido um lançamento interno do sistema em maio de 1995" (BASSO, 2014, p. 196). Ele é um dos sistemas de gerenciamento de banco de dados mais utilizados no mundo.

A integração entre o PHP e o MySQL é amplamente utilizada em aplicações *web* baseadas em PHP, como WordPress, Joomla, Drupal, Magento, entre outras. Essas aplicações usam o MySQL para armazenar e gerenciar dados, como conteúdo de páginas, informações de usuário e produtos em lojas virtuais.

Em resumo, a integração entre o PHP e o MySQL é uma combinação poderosa e popular para o desenvolvimento de aplicações *web*. Com a sua facilidade de uso, suporte a várias plataformas e linguagem SQL, o PHP e o MySQL se tornaram uma dupla confiável para armazenar, gerenciar e recuperar dados em aplicações *web* baseadas em PHP.

MySQL

O MySQL é um sistema de gerenciamento de banco de dados relacional de código aberto amplamente utilizado para armazenar, organizar e gerenciar dados em aplicações *web*. Ele suporta a linguagem SQL (Structured Query Language), que é usada para realizar operações em bancos de dados, como inserir, atualizar e recuperar dados, sendo capaz de gerenciar grandes quantidades de dados de forma eficiente e escalável (BASSO, 2014). O MySQL é compatível com várias plataformas e pode ser integrado com várias linguagens de programação, incluindo PHP, Python e Java. Ele também oferece recursos avançados, como replicação, *clustering* e particionamento, permitindo que os desenvolvedores configurem o MySQL para atender às necessidades específicas de suas aplicações.

Os bancos de dados são essenciais para muitas aplicações *web*, pois são onde a aplicação armazena e gerencia os dados essenciais para o seu fun-

cionamento. O MySQL é um dos bancos de dados mais populares do mundo e é amplamente usado em aplicações *web* baseadas em PHP.

Conexão do MySQL à aplicação PHP

Agora que você já conhece a importância da conexão entre o PHP e um banco de dados MySQL para o desenvolvimento de sistemas *web*, é possível aprender mais sobre as principais formas de estabelecer essa conexão. Para acessar um banco de dados MySQL pelo PHP, é necessário estabelecer uma conexão entre o sistema *web* e o banco de dados.

Saiba mais

Em PHP, é possível importar funções e variáveis definidas em outros arquivos para o arquivo atual, como o arquivo com os dados de conexão, por meio de funções de importação, como `include()` e `require()`. A diferença entre elas é que a `include()` apenas emite um aviso caso o arquivo não exista, enquanto a `require()` emite um erro fatal. Além disso, existem as funções `include_once` e `require_once`, que evitam a reimportação do mesmo arquivo caso ele já tenha sido incluído anteriormente. Essas funções são úteis para manter o código organizado e facilitar a manutenção.

Recomenda-se a criação de um arquivo com os dados de conexão e a importação dele nos locais necessários para acessar o banco de dados MySQL pelo PHP. Essa abordagem evita a repetição de informações e facilita a manutenção do código, além de permitir a reutilização do mesmo arquivo de conexão em diferentes projetos. Com isso, a conexão é estabelecida por meio da passagem de informações que permitem a atribuição de acesso de acordo com os privilégios de cada aplicação.

Para realizar essa conexão, as duas principais opções em PHP são a extensão `mysqli` (MySQL Improved) e o PHP Data Objects (PDO). Ambas oferecem recursos e funcionalidades para conectar, gerenciar e manipular dados no banco de dados MySQL, permitindo o desenvolvimento de aplicações *web* em PHP.

Fique atento

Existe uma terceira forma de conexão com o banco de dados MySQL: a extensão MySQL, que está obsoleta e não deve ser mais utilizada, porém ainda existem sistemas legados que a utilizam (CHAIBEN, 2015). Assim, as três principais extensões para conexão com o banco de dados MySQL em PHP são: MySQL (obsoleta desde o PHP 5.5.0), MySQLi (melhorada em relação à extensão MySQL) e PDO (que suporta vários bancos de dados, incluindo MySQL).

Extensão `mysqli`

A extensão `mysqli` é uma das principais formas de conectar o PHP ao banco de dados MySQL (BASSO, 2014). Ela oferece uma série de recursos para realizar a conexão, como a possibilidade de executar consultas SQL, recuperar resultados e manipular dados. Além disso, a extensão `mysqli` é altamente eficiente e escalável, permitindo lidar com grandes volumes de dados sem comprometer a *performance* da aplicação.

É importante destacar que a extensão `mysqli` também oferece recursos de segurança, como a possibilidade de preparar consultas para evitar ataques de injeção de SQL e a utilização de transações para garantir a integridade dos dados (BIERER; HUSSAIN; AJZELE, 2016). Assim, a extensão `mysqli` é uma excelente opção para conectar o PHP ao MySQL e desenvolver sistemas *web* de alta qualidade.

A extensão `mysqli` oferece uma série de comandos para conectar o PHP ao MySQL e realizar a manipulação de dados. A seguir, veja alguns dos principais comandos (ABT *et al*., c2023).

- `mysqli _ connect()`: utilizado para realizar a conexão com o banco de dados MySQL.
- `mysqli _ mysqli _ close()`: serve para executar consultas SQL no banco de dados.
- `mysqli _ fetch _ array()`: usado para recuperar resultados de consultas SQL em formato de *array*.
- `mysqli _ num _ rows()`: serve para contar o número de linhas retornadas por uma consulta SQL.
- `mysqli _ affected _ rows()`: utilizado para obter o número de linhas afetadas por uma operação (inserção, atualização ou exclusão).
- `mysqli _ real _ escape _ string()`: usado para escapar caracteres especiais em *strings* que serão utilizadas em consultas SQL, a fim de evitar ataques de injeção de SQL.

Esses são apenas alguns exemplos dos comandos disponíveis na extensão `mysqli`. Há diversos outros comandos e recursos que permitem conectar, gerenciar e manipular dados no banco de dados MySQL a partir de uma aplicação *web* desenvolvida em PHP.

A princípio, precisamos realizar a conexão entre a aplicação que estamos desenvolvendo e o banco de dados. Para tanto, observe um exemplo de forma de conexão utilizando a extensão `mysqli`:

```
<?php

// Configurações de conexão com o banco de dados
$servidor = "localhost";
$usuario = "usuario _ do _ banco";
$senha = "senha _ do _ banco";
$banco = "nome _ do _ banco";

// Criação da conexão
$conexao = mysqli _ connect($servidor, $usuario, $senha,
$banco);

// Verificando se a conexão foi bem-sucedida
if (!$conexao) {
    die("Conexão falhou: " . mysqli _ connect _ error());
}

echo "Conexão bem-sucedida";

// Fechamento da conexão
mysqli _ close($conexao);

?>
```

Nesse exemplo, são definidas as informações necessárias para conectar ao banco de dados MySQL, como o nome do servidor, o nome de usuário,

a senha e o nome do banco de dados (BASSO, 2014). Em seguida, é criada a conexão utilizando a função `mysqli _ connect()`. Caso a conexão seja bem-sucedida, a mensagem "Conexão bem-sucedida" é exibida. Por fim, a conexão é fechada utilizando a função `mysqli _ close()`. Esse é apenas um exemplo básico, mas é possível utilizar a extensão `mysqli` para executar consultas SQL, recuperar resultados e manipular dados de diversas formas.

Também é possível realizar a conexão orientada a objetos:

```
$conexao = new mysqli($servidor, $usuario, $senha, $banco);
```

A diferença entre as duas formas de conexão é basicamente a sintaxe utilizada. Na primeira forma, utiliza-se a função `new mysqli()` para criar um objeto `mysqli` que representa a conexão com o banco de dados. Os parâmetros passados são o nome do servidor, o nome de usuário, a senha e o nome do banco de dados (ABT *et al*., c2023). Já na segunda forma, utiliza-se a função `mysqli _ connect()` para estabelecer a conexão com o banco de dados. Os parâmetros são os mesmos da primeira forma, porém, ao contrário da primeira, essa função retorna uma conexão com o banco de dados, não um objeto `mysqli`.

Ambas as formas são válidas e utilizadas no PHP, e a escolha de qual usar depende do desenvolvedor e da equipe que está trabalhando no projeto. Há quem prefira a primeira forma, por ser mais orientada a objetos, e há quem prefira a segunda, por ser mais simples e direta.

PDO

O PDO é uma extensão do PHP que oferece uma camada de abstração para conexão com bancos de dados. Essa extensão permite que os desenvolvedores trabalhem com diversos tipos de bancos de dados, não apenas MySQL, utilizando uma interface unificada para executar consultas e manipular dados (BASSO, 2014).

O PDO é uma solução popular para quem precisa trabalhar com bancos de dados em PHP, pois oferece uma série de recursos e benefícios, como a possibilidade de trabalhar com vários bancos de dados em uma mesma aplicação, o suporte a transações e a possibilidade de utilizar *prepared statements*, que ajudam a prevenir ataques de injeção de SQL (ABT *et al*., c2023).

Para utilizar o PDO, é necessário criar uma instância da classe PDO e passar como parâmetros as informações de conexão com o banco de dados, como o nome do servidor, o nome de usuário, a senha e o nome do banco de dados. A partir dessa instância, é possível executar consultas SQL com a função `mysqli _ close()` ou usar *prepared statements* com a função `prepare()`.

O PDO também oferece uma série de métodos para recuperar resultados e manipular dados de forma segura e eficiente.

A seguir, vamos analisar como realizar a conexão com o banco de dados, agora utilizando a extensão PDO.

```
<?php

// Informações de conexão com o banco de dados

$servidor = "localhost";

$usuario = "usuario _ do _ banco";

$senha = "senha _ do _ banco";

$banco = "nome _ do _ banco";


// Criando uma instância do PDO

try {

    $conexao=new PDO("mysql:host=$servidor;dbname=$banco;c
harset=utf8", $usuario, $senha);

} catch (PDOException $e) {

     echo 'Erro ao conectar com o banco de dados: ' .
$e->getMessage();

    exit();

}


// Executando uma consulta simples

$sql = "SELECT * FROM tabela";

$stmt = $conexao->query($sql);


// Iterando sobre os resultados

while ($row = $stmt->fetch(PDO::FETCH _ ASSOC)) {
```

```
    echo $row['campo1'] . ', ' . $row['campo2'] . '<br>';
}

// Fechando a conexão
$conexao = null;
?>
```

Nesse exemplo, é criada uma instância do PDO utilizando as informações de conexão com o banco de dados. Em seguida, é executada uma consulta SQL simples utilizando a função `mysqli _ close()`, e os resultados são recuperados utilizando o método `fetch()`. Por fim, a conexão com o banco de dados é fechada utilizando a função `null`. Note que esse é apenas um exemplo básico; existem muitas outras funcionalidades disponíveis no PDO.

Nesta seção, conhecemos as principais formas de conexão do PHP com o MySQL e vimos exemplos de códigos para realizar a conexão com o banco de dados. É importante ressaltar que também existe a extensão MySQL, embora ela esteja obsoleta e não deva mais ser utilizada. Na próxima seção, vamos explorar os principais comandos de manipulação de dados no banco via PHP, desde a inserção de novos dados até a exclusão de registros existentes.

Manipulação de dados com PHP

Agora que você já sabe como conectar a sua aplicação ao banco de dados, é hora de aprender a manipular os dados armazenados nele. A manipulação de dados é essencial para qualquer sistema para internet, já que permite que informações sejam armazenadas, consultadas, atualizadas e excluídas de forma eficiente e segura. Em um sistema *web*, os dados podem estar relacionados a diversos aspectos, como cadastro de usuários, informações de produtos, histórico de compras, entre outros.

No contexto do SQL, existem quatro principais operações de manipulação de dados: `INSERT`, `SELECT`, `UPDATE` e `DELETE`. O comando `INSERT` é utilizado para adicionar novas informações em uma tabela do banco de dados, já o `SELECT` é utilizado para consultar e exibir dados já existentes. Por sua vez, o comando `UPDATE` permite a alteração de informações já existentes em uma tabela, e o `DELETE` é utilizado para excluir informações

de uma tabela. Esses comandos podem ser utilizados de forma combinada e personalizada, a fim de atender às necessidades específicas de cada sistema (BASSO, 2014).

Fique atento

Para evitar erros e vulnerabilidades é essencial adotar boas práticas, como a validação de dados (BIERER; HUSSAIN; AJZELE, 2016). Além disso, é importante ter cautela ao manipular informações sensíveis, como dados de clientes e transações financeiras.

Entre as quatro operações básicas, três delas são capazes de alterar os dados armazenados no banco: `INSERT`, `UPDATE` e `DELETE`. É importante ressaltar que essas operações devem ser executadas com cautela, já que podem impactar na integridade e na segurança dos dados armazenados (BIERER; HUSSAIN; AJZELE, 2016). A seguir, vamos explorar esses três comandos.

Comando `INSERT`

O comando `INSERT` é utilizado para adicionar novas informações em uma tabela do banco de dados (BASSO, 2014). A sintaxe básica do comando é simples: basta especificar o nome da tabela e os valores que serão inseridos em cada coluna correspondente. É importante lembrar que os valores devem estar de acordo com o tipo de dados definido para cada coluna.

A sintaxe básica do comando `INSERT` em SQL é (NIXON, 2021):

```
INSERT INTO tabela (coluna1, coluna2, ...)

VALUES (valor1, valor2, ...)
```

onde:

- `tabela` é o nome da tabela na qual os novos registros serão inseridos;
- `coluna1, coluna2, ...` são as colunas nas quais os novos valores serão inseridos;
- `valor1, valor2, ...` são os valores que serão inseridos nas colunas especificadas.

Agora vamos aplicar essa estrutura. Observe o seguinte exemplo de uma inserção de dados:

```
INSERT INTO clientes (nome, email, telefone) VALUES ('João',
'joao@email.com', '555-1234');
```

Nesse exemplo, estamos inserindo um novo registro na tabela `clientes`, com os valores `João` para o nome, `joao@email.com` para *e-mail* e `555-1234` para telefone. Para realizar a inserção por meio do PHP, podemos utilizar uma das extensões apresentadas anteriormente.

Observe no exemplo a seguir como realizar a inserção utilizando a extensão `mysqli`:

```
// Configurações de conexão com o banco de dados

$servidor = "localhost";

$usuario = "usuario _ do _ banco";

$senha = "senha _ do _ banco";

$banco = "nome _ do _ banco";


// Criação da conexão

$conexao = mysqli _ connect($servidor, $usuario, $senha,
$banco);


// Verificando se a conexão foi bem-sucedida

if (!$conexao) {

    die("Conexão falhou: " . mysqli _ connect _ error());

}


// Definição da query de inserção

$sql = "INSERT INTO clientes (nome, email, telefone) VALUES
('João', 'joao@email.com', '123456789')";


// Execução da query
```

```
if (mysqli_query($conexao, $sql)) {
    echo "Dados inseridos com sucesso!";
} else {
   echo "Erro ao inserir dados: " . mysqli_error($conexao);
}

// Fechamento da conexão
mysqli_close($conexao);
```

Nesse exemplo, inserimos dados na tabela `clientes` do banco de dados e definimos os valores para os campos `nome`, `email` e `telefone`. Em seguida, executamos a *query* utilizando a função `mysqli_mysqli_close()`, que retorna verdadeiro se a operação for bem-sucedida e falso caso contrário. Por fim, fechamos a conexão com o banco de dados utilizando a função `mysqli_close()`.

Comando `UPDATE`

O comando `UPDATE`, como já mencionado, é utilizado para alterar informações já existentes em uma tabela do banco de dados. É comum utilizar esse comando para corrigir erros em dados inseridos anteriormente, atualizar informações de registros específicos ou aplicar alterações em massa em uma tabela (BASSO, 2014).

Para realizar um `UPDATE`, é necessário especificar a tabela a ser alterada, as colunas que serão atualizadas e as condições para identificar quais registros serão afetados. É importante definir com precisão as condições para evitar alterações indesejadas em registros que não deveriam ser modificados.

A sintaxe básica do comando `UPDATE` em SQL é a seguinte (NIXON, 2021):

```
UPDATE tabela
SET coluna1 = valor1, coluna2 = valor2, ...
WHERE condição
```

onde:

- `tabela` é o nome da tabela que será atualizada;
- `coluna1, coluna2, ...` são as colunas que serão atualizadas;
- `valor1, valor2, ...` são os novos valores que serão atribuídos às colunas especificadas;
- `condição` é a cláusula que define quais registros serão afetados.

Agora vamos aplicar essa estrutura. Observe o seguinte exemplo de uma atualização de dados.

```
UPDATE produtos SET nome = 'Novo nome do produto' WHERE
id = 1;
```

Nesse exemplo, a coluna `nome` receberá o valor `João`, e a coluna `email` receberá o valor `joao@email.com`. Por fim, a cláusula `WHERE` é utilizada para especificar qual registro deve ser atualizado. Nesse caso, apenas o registro com o valor 1 na coluna ID será atualizado. No exemplo, estamos atualizando o registro do cliente com ID igual a 1. A cláusula `WHERE` é importante para especificar qual registro será atualizado. Sem ela, todos os registros da tabela seriam afetados.

A seguir, confira um novo exemplo e veja como realizar a atualização da sua aplicação PHP utilizando a extensão `mysqli`.

```
<?php

// Configurações de conexão com o banco de dados

$servidor = "localhost";

$usuario = "usuario _ do _ banco";

$senha = "senha _ do _ banco";

$banco = "nome _ do _ banco";


// Criação da conexão

$conexao = mysqli _ connect($servidor, $usuario, $senha,
$banco);


// Verificando se a conexão foi bem sucedida

if (!$conexao) {
```

```
    die("Conexão falhou: " . mysqli_connect_error());
}

// Query para atualização
$sql = "UPDATE clientes SET nome='João', email='joao@email.
com'
        WHERE id=1";

// Definição da query de atualização
if (mysqli_query($conexao, $sql)) {
    echo "Registro atualizado com sucesso!";
} else {
      echo "Erro na atualização do registro: " .
mysqli_error($conexao);
}

// Fechando a conexão
mysqli_close($conexao);
?>
```

Antes de passarmos para o próximo comando, vale ressaltar que a cláusula `WHERE`, mesmo sendo opcional, tem grande importância em uma atualização de dados, pois, como já dito, se ela não for especificada todos os registros da tabela serão atualizados. Lembre-se de utilizá-la sempre que possível. Uma boa prática em programação é utilizar a cláusula `WHERE` para especificar somente os registros que se deseja atualizar, evitando atualizações desnecessárias e indesejadas.

Comando DELETE

O comando DELETE é utilizado para excluir registros de uma tabela no banco de dados. Assim como no comando UPDATE, a cláusula WHERE é utilizada para especificar quais registros devem ser excluídos. Se a cláusula WHERE não for especificada, todos os registros da tabela serão excluídos (BASSO, 2014).

Assim como na inserção e na atualização de dados, é importante tomar cuidado ao utilizar o comando DELETE, pois a exclusão de dados pode ser irreversível e afetar a integridade do sistema. Por isso, é essencial ter certeza de que os registros a serem excluídos são os corretos e que a ação não vai prejudicar outras partes do sistema.

Fique atento

Em alguns sistemas, em vez de excluir um registro, é comum utilizar uma exclusão lógica, incluindo uma coluna na tabela para indicar que o registro está inativo ou excluído. Assim, o registro ainda permanece na tabela, mas não é mais visível nas consultas. Essa prática pode ser útil para manter um histórico de ações ou evitar a exclusão acidental de registros importantes.

A sintaxe básica do comando DELETE no SQL é a seguinte (NIXON, 2021).

```
DELETE FROM nome_da_tabela WHERE condição;
```

onde:

- nome_da_tabela é o nome da tabela da qual deseja-se excluir registros;
- condição é a cláusula opcional que especifica qual registro deve ser excluído. Se não for especificada, todos os registros da tabela serão excluídos.

É importante ressaltar que o comando DELETE pode ser perigoso se não utilizado corretamente, pois pode resultar na exclusão permanente de dados. Por isso, é recomendado sempre fazer um *backup* dos dados antes de executar o comando.

Observe o exemplo:

```
DELETE FROM clientes

WHERE id = 1;
```

Esse comando vai excluir o registro da tabela `clientes` que tem o ID igual a 1. É importante ressaltar que a cláusula `WHERE` é utilizada para especificar qual registro deve ser excluído.

Agora observe, no exemplo a seguir, como realizar a atualização com a aplicação PHP, utilizando a extensão `mysqli`.

```
<?php


// Configurações de conexão com o banco de dados

$servidor = "localhost";

$usuario = "usuario _ do _ banco";

$senha = "senha _ do _ banco";

$banco = "nome _ do _ banco";


// Criação da conexão

$conexao = mysqli _ connect($servidor, $usuario, $senha,
$banco);


// Verificando se a conexão foi bem sucedida

if (!$conexao) {

    die("Conexão falhou: " . mysqli _ connect _ error());

}


// Execução da exclusão de um registro na tabela "clientes"

$sql = "DELETE FROM clientes WHERE id=1";


if ($conn->query($sql) === TRUE) {

  echo "Registro excluído com sucesso";
```

```
} else {
  echo "Erro ao excluir registro: " . $conn->error;
}

// Fechando a conexão
$conn->close();

?>
```

Nesse exemplo, estamos excluindo o registro com ID igual a 1 na tabela `clientes`. Note que o processo é similar aos exemplos de inserção e de atualização, com a diferença de que estamos utilizando o comando `DELETE` em vez de `INSERT` ou `UPDATE`.

Agora que você já conhece os três comandos básicos de manipulação de dados, vamos abordar o comando `SELECT`, que permite a seleção e a consulta de dados em um banco de dados.

Seleção de dados com `SELECT`

O comando `SELECT` é usado para buscar dados em uma ou mais tabelas de um banco de dados relacional. Ele é um dos comandos mais utilizados na linguagem SQL e permite realizar consultas específicas para selecionar, filtrar e ordenar os dados desejados (BASSO, 2014).

Basicamente, o comando `SELECT` permite selecionar quais colunas de uma tabela devem ser recuperadas e quais condições devem ser aplicadas para filtrar os dados. A sintaxe básica do comando `SELECT` é a seguinte (NIXON, 2021).

```
SELECT coluna1, coluna2, ...
FROM tabela
WHERE condição
```

onde:

- `coluna1, coluna2, ...` são as colunas que se deseja recuperar;
- `tabela` é o nome da tabela da qual deseja-se recuperar os dados;
- `condição` é uma condição opcional que deve ser satisfeita pelos dados para serem retornados.

As cláusulas mais importantes do comando `SELECT` são as seguintes.

- `WHERE`: permite especificar uma ou mais condições para filtrar os dados a serem retornados. As condições podem ser expressas em termos de operadores lógicos (`AND`, `OR`) e operadores relacionais (>, <, =, <>).
- `ORDER BY`: permite ordenar os resultados de acordo com uma ou mais colunas em ordem crescente ou decrescente.
- `GROUP BY`: permite agrupar os resultados de acordo com uma ou mais colunas. Geralmente é utilizada com funções de agregação, como `SUM`, `AVG`, `COUNT`, entre outras.
- `HAVING`: permite filtrar os resultados obtidos após a aplicação da cláusula `GROUP BY`, utilizando condições semelhantes às cláusulas `WHERE`.

Essas cláusulas permitem que o comando `SELECT` seja utilizado de forma mais poderosa e eficiente para a manipulação de dados em SQL. Veja a seguir alguns exemplos de ações possíveis com o comando `SELECT`.

```
//Selecionar todos os registros de uma tabela:

SELECT * FROM tabela;


//Selecionar registros específicos de uma tabela com base
em uma condição:

SELECT * FROM tabela WHERE coluna = valor;


//Selecionar registros de uma tabela em ordem crescente:

SELECT * FROM tabela ORDER BY coluna ASC;


//Selecionar registros de uma tabela em ordem decrescente:

SELECT * FROM tabela ORDER BY coluna DESC;
```

```
//Selecionar registros de uma tabela com um limite máximo
de resultados:

SELECT * FROM tabela LIMIT 10;


//Selecionar registros de uma tabela com um limite máximo de
resultados e um offset (pular um certo número de resultados):

SELECT * FROM tabela LIMIT 10 OFFSET 20;


//Selecionar registros de várias tabelas com base em uma
condição de junção:

SELECT * FROM tabela1 INNER JOIN tabela2 ON tabela1.coluna
= tabela2.coluna;
```

Esses são apenas alguns exemplos de comandos `SELECT` em SQL. Existem muitas outras variações e opções disponíveis, dependendo das necessidades específicas do projeto.

Para utilizar o comando `SELECT` no PHP, é necessário primeiro estabelecer uma conexão com o banco de dados utilizando as funções do PHP para manipulação de banco de dados, como `mysqli _ connect()` ou PDO. Em seguida, é possível executar uma *query* `SELECT` utilizando a função `mysqli _ mysqli _ close()` ou `PDO::mysqli _ close()` e passando como argumento a *query* SQL desejada. O resultado da *query* pode ser armazenado em uma variável, com uso da função `mysqli _ fetch _ assoc()` ou `PDO::fetch()` para obter cada linha de resultado.

A seguir, veja um exemplo de código PHP, utilizando a extensão `mysqli`, que realiza uma consulta `SELECT` na tabela `clientes` e exibe o resultado na tela.

```
<?php

// Configurações de conexão com o banco de dados

$servidor = "localhost";

$usuario = "usuario _ do _ banco";

$senha = "senha _ do _ banco";

$banco = "nome _ do _ banco";
```

```
// Criação da conexão
$conexao = mysqli _ connect($servidor, $usuario, $senha,
$banco);

// Verificando se a conexão foi bem-sucedida
if (!$conexao) {
    die("Conexão falhou: " . mysqli _ connect _ error());
}

// Execução da consulta SELECT
$sql = "SELECT * FROM clientes";
$resultado = mysqli _ query($conexao, $sql);

// Exibindo o resultado na tela
if (mysqli _ num _ rows($resultado) > 0) {
  while ($row = mysqli _ fetch _ assoc($resultado)) {
    echo "ID: " . $row["id"] . " - Nome: " . $row["nome"] .
" - Email: " . $row["email"] . "<br>";
  }
} else {
  echo "Nenhum resultado encontrado.";
}

// Fechando a conexão com o banco de dados
mysqli _ close($conn);
?>
```

Nesse exemplo, a *query* `SELECT` seleciona todos os campos da tabela `clientes`. A função `mysqli_fetch_assoc()` é utilizada para obter cada linha de resultado como um *array* associativo, que então é usado para exibir os dados na tela. Aqui também podemos observar o comando `mysqli_num_rows`, usado para verificar se a consulta retornou algum resultado. Se o número de linhas retornadas for maior que zero, o código vai entrar no bloco `if`. Dentro do bloco `if`, um *loop* `while` é usado para percorrer os resultados da consulta. Para cada linha encontrada, o código imprime o ID, o nome e o *e-mail* do registro. Caso não seja retornado um resultado, o código vai entrar no bloco `else` e vai imprimir a mensagem "Nenhum resultado encontrado".

Neste capítulo, vimos o conceito de banco de dados e a sua importância na programação *web*. Além disso, estudamos como integrar o PHP com o banco de dados e quais são os três principais comandos utilizados: `INSERT`, `UPDATE` e `DELETE`. Para realizar consultas, também estudamos o comando `SELECT`, suas cláusulas mais importantes e como utilizá-lo tanto em SQL quanto em PHP. A compreensão desses tópicos é fundamental para qualquer desenvolvedor *web*, sendo uma base sólida para aprender conceitos mais avançados no futuro.

Referências

ABT, B. *et al*. Manual do PHP. *PHP Documentation Group*, c2023. Disponível em: https://www.php.net/manual/pt_BR/index.php. Acesso em: 23 mar. 2023.

BASSO, L. O. Integração de PHP e MySQL. *In*: MILETTO, E. M.; BERTAGNOLLI, S. C. (org.). *Desenvolvimento de software II*. Porto Alegre: Bookman, 2014. p. 195-212.

BIERER, D.; HUSSAIN, A.; AJZELE, B. *PHP 7*: real world application development. Birmingham: Packt, 2016. *E-book*.

CHAIBEN, R. B. MySQL obsoleto: não utilize funções MySQL. *IMasters*, 2015. Disponível em: https://imasters.com.br/back-end/mysql-obsoleto-nao--utilize-funcoes-mysql_. Acesso em: 23 mar. 2023.

NIXON, R. *Learning PHP, MySQL & JavaScript*: a step-by-step guide to creating dynamic websites. 6th ed. Sebastopol: O'Reilly, 2021.

Leituras recomendadas

OLSSON, M. *PHP 7 quick scripting reference*. 2nd ed. New York: Springer, 2016.

PRETTYMAN, S. *Learn PHP 7*. New York: Springer, 2016. *E-book*.

SOARES, W. *PHP 5*: conceitos, programação e integração com banco de dados. 7. ed. São Paulo: Érica, 2013.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Conteúdo:

sagah

Dica do Professor

Nos dias atuais, as aplicações *web* dinâmicas têm a necessidade de se integrar com um banco de dados para todas as finalidades, seja para o armazenamento de usuários e senhas, seja para gerenciar *websites* complexos, como os portais de notícias e as lojas virtuais. Ao usarmos o PHP com o SGBD MySQL, podemos verificar o poder da integração da linguagem de programação com os recursos de manipulação de banco de dados.

No vídeo desta Dica do Professor, você vai saber como se faz a integração entre PHP e o SGBD MySQL.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

1) **O PHP dispõe de ferramentas para acesso a dados que permitem uma conexão com bancos de dados, como o MySQL.**

Quais são as três extensões do PHP para conexão no banco de dados MySQL?

A) Include, Require e Session.

B) SQL, CSS e Javascript.

C) SQL, PDO e SESSION.

D) MySQL, MySQLi e PDO.

E) MySQLi, PDO e mysqli_fecth_array.

2) **Em PHP, é possível reutilizar código de outros arquivos/ *scripts*.**

Quais comandos do PHP são utilizados para esse fim?

A) Include, Require e mysqli_num_rows.

B) If, Else e Mysql_assoc.

C) Include, Include_once, Require e Require_once.

D) Require_once, For e While.

E) Array, Class e Função.

3) **MySQLi é uma biblioteca para conexão ao MySQL, que foi disponibilizada a partir do PHP5.**

Qual é o comando para verificar o número de linhas encontradas em uma consulta SQL realizada com a biblioteca MySQLi?

A) "mysqli_fecth_array".

B) "mysqli_num_rows".

C) "Session_start();".

D) "Select * from produtos".

E) "mysql_result();"

4) **MySQLi é uma das maneiras de acessar um servidor de banco de dados MySQL.**

Para criarmos uma conexão com o banco de dados utilizando a biblioteca MySQLi, usamos a expressão:

A) $conn = mysql connect "servidor, usuario, senha, banco_de_dados";

B) public static java.sql.Connection getConexaoMySQL()

C) $conn = new PDO("servidor, usuario, senha, banco_de_dados");

D) $conn = new mysqli(servidor, usuario, senha, banco_de_dados);

E) $conn->close;

5) **O MySQLi serve para aproveitarmos os recursos do servidor MySQL, como a realização de consultas.**

Para fazer uma consulta simples com PHP e MySQLi, usamos a expressão:

A) $sql = "UPDATE banco set *";

B) $sql = "SELECT * FROM banco";

C) $sql = "DELETE FROM banco";

D) $sql = "CREATE TABLE banco";

E) session_start();

Na prática

Desde o início, a linguagem PHP sempre manteve como “parceiros” o Apache como Servidor Web e o MySQL como SGBD. O Servidor Web tem a função de fazer com que a página *web* dinâmica seja processada retornando a informação necessária, ao passo que o SGBD MySQL funciona do lado do servidor e tem como função o gerenciamento dos dados.

Acompanhe este Na Prática, que mostra como a integração da linguagem PHP com o MySQL pode ser muito eficiente para excluir, incluir e alterar informações em banco de dados.

INTEGRAÇÃO DA LINGUAGEM PHP COM O MYSQL

PROBLEMA

O gestor de desenvolvimento de uma indústria de cosméticos e sua equipe precisam, constantemente, atualizar valores e excluir produtos que não estão mais no catálogo da loja virtual da empresa.

MOTIVO

Produtos deixam de ser fabricados, preços sofrem alterações, novos produtos são desenvolvidos, certos produtos não passam pelo selo de qualidade, etc.

SOLUÇÃO

Então, para que todas essas alterações sejam realizadas de forma satisfatória, os desenvolvedores precisam utilizar a integração do PHP com o banco de dados MySQL, e assim todas as tarefas serão executadas de forma prática e segura.

Base de dados MySQL

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

Sistema de *login* com PHP e MySQL

Acesse o vídeo e veja um tutorial sobre como criar um sistema de *login* com PHP e MySQL.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

PHP e MYSQL: conectando e exibindo dados de forma rápida – Dica

Veja, neste artigo, como criar uma conexão com banco de dados MySQL utilizando PHP e como exibir os dados cadastrados em uma tabela.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Desenvolvimento *web* com PHP

Leia este material e aprenda um pouco mais sobre a linguagem PHP, sua sintaxe, uso de funções, orientação a objetos, sua integração do MySQL, entre outros recursos.

Conteúdo interativo disponível na plataforma de ensino!