PHP - MySQLi

Interazione tra PHP e Database MySQL

Fedeli Massimo

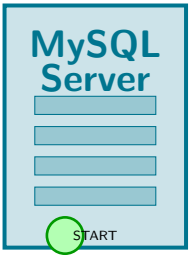
IIS E.Fermi - Sacconi - Ceci

Indice

Il DBMS MySQL

MySQL è un DBMS Server con le seguenti caratteristiche:

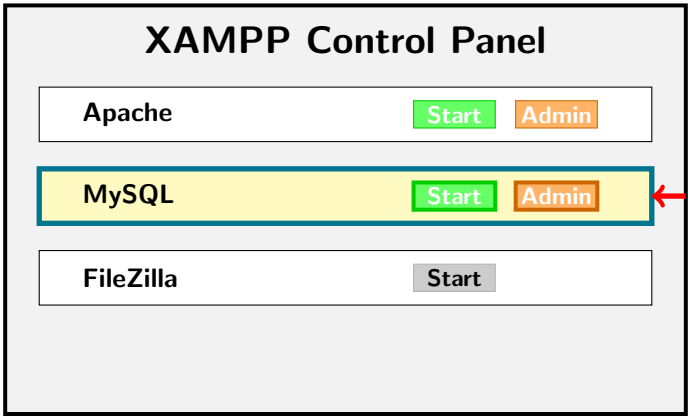
- Basato su SQL
- Multipiattaforma
- Relazionale
- Licenza open source



Avvio del servizio:

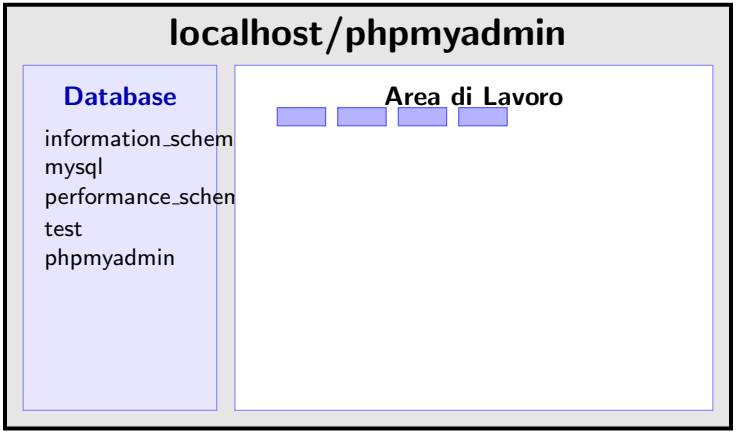
1. Aprire il pannello XAMPP
2. Fare clic su "Start" accanto a MySQL
3. Accedere a phpMyAdmin tramite "Admin"

Pannello di controllo XAMPP



phpMyAdmin - Ambiente di Amministrazione

- **phpMyAdmin** è l'ambiente di amministrazione grafico per MySQL
- Permette di gestire database e tabelle tramite interfaccia GUI
- Accessibile tramite browser web

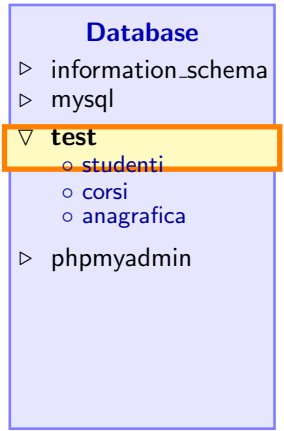


Selezione Database in phpMyAdmin

Operazioni disponibili:

1. Selezionare un database dalla colonna sinistra
2. Espandere il simbolo "+" per vedere le tabelle
3. Fare clic sul nome per accedere alle funzioni

Esempio: database `test`



La Classe MySQLi

Caratteristiche Principali

PHP dispone della classe `mysqli` per l'interazione con il DBMS MySQL

Approccio Procedurale

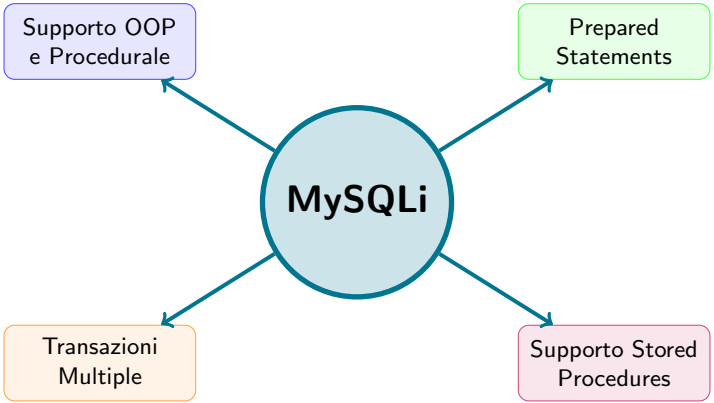
Funzioni tradizionali
mysqli_connect()
mysqli_query()

Approccio Object-Oriented

Metodi di classe
$conn = new mysqli()
$conn-¿query()

I due approcci possono essere utilizzati nello stesso script!

Interfaccia MySQLi - Vantaggi



Connessione al Database - Sintassi

Sintassi Object-Oriented

```
1 $conn = new mysqli($nomeHost, $username, $password, $dbname);
```

Parametri:

$nomeHost	Macchina che esegue il DBMS (es. "localhost")
$username	Utente per l'accesso al database (es. "root")
$password	Password associata all'utente
$dbname	Nome del database da utilizzare

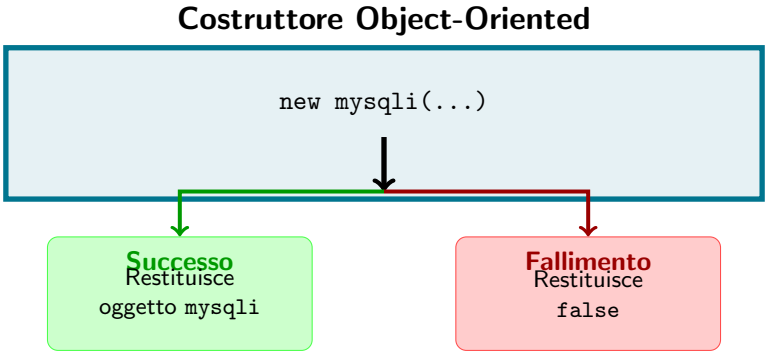
Importante

La variabile `$conn` è l'oggetto che rappresenta la connessione al database

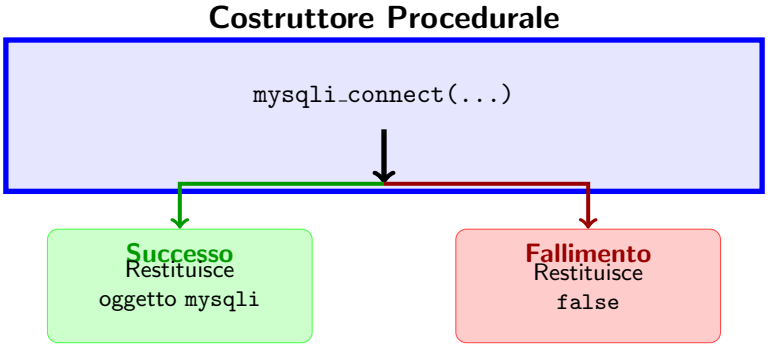
Esempio di Connessione

```php
<?php
// Parametri di connessione
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "test";

// Creazione connessione
$conn = new mysqli($servername, $username, $password, $dbname);

// Verifica connessione
if ($conn->connect_error) {
    die("Connessione fallita: " . $conn->connect_error);
}
echo "Connessione riuscita";
?>
```

Valori Restituiti - Stile OOP



Valori Restituiti - Stile Procedurale



Il Metodo connect_errno

Descrizione

Il metodo `connect_errno()` verifica se la connessione è avvenuta regolarmente

Valore di ritorno:

- 0 → Connessione riuscita
- `Numero diverso da 0` → Codice di errore

```php
if ($conn->connect_errno) {
    echo "Errore connessione: " . $conn->connect_errno;
    exit();
}
```

https://www.w3schools.com/php/func_mysqli_connect_errno.asp

Gestione Errori di Connessione

Metodi per gestire gli errori:

- `connect_errno`: codice errore
- `connect_error`: descrizione errore

Fatal Error

Access denied for
user 'root'@'localhost'
(using password: YES)

```php
if ($conn->connect_error) {
    die("Connessione fallita: " . $conn->connect_error);
}
```

https://www.w3schools.com/php/func_mysqli_connect_error.asp

Chiusura della Connessione

Importante!

È fondamentale chiudere la connessione al termine dello script per:

- Liberare risorse del server
- Evitare sovraccarico del Web server
- Prevenire problemi di sicurezza

```php
// Chiusura della connessione
$conn->close();
```

https://www.php.net/manual/en/mysqli.close.php

Note sulla Chiusura - Documentazione Ufficiale

Dalla documentazione PHP

"Open non-persistent MySQL connections and result sets are automatically closed when their objects are destroyed. Explicitly closing open connections and freeing result sets is optional."

Best Practice

"However, it's a good idea to close the connection as soon as the script finishes performing all of its database operations, if it still has a lot of processing to do after getting the results."

`https://www.php.net/manual/en/mysqli.close.php`

Il Metodo query()

Descrizione

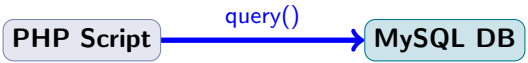
Il metodo `query()` invia una query SQL al database aperto

Sintassi:

```php
$result = $conn->query($sql);
```

Esempio:

```php
$sql = "SELECT * FROM Anagrafica WHERE Cognome LIKE 'Ros%'";
$ris = $conn->query($sql);
```

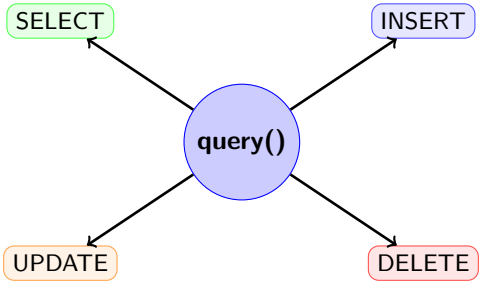


Metodo query() - Parametri

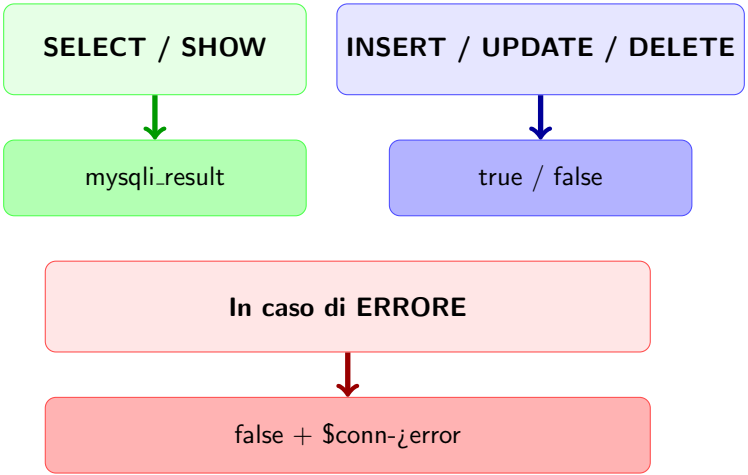
Parametri in Ingresso

$query (string) La stringa contenente la query SQL da eseguire

Tipologie di query supportate:



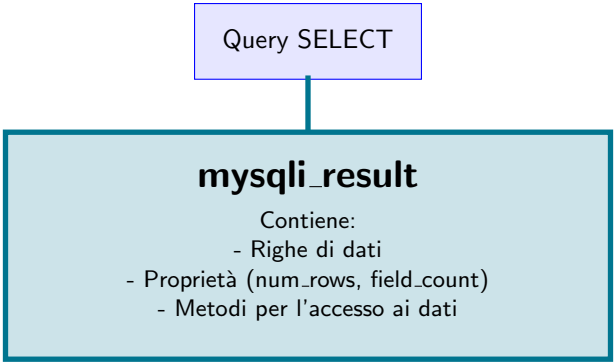
Metodo query() - Valori Restituiti



L'Oggetto mysqli_result

Definizione

L'oggetto `mysqli_result` rappresenta il risultato di una query eseguita su MySQL tramite l'estensione mysqli di PHP



Struttura mysqli_result

mysqli_result

Proprietà Principali

`num_rows`: numero di righe nel risultato
`field_count`: numero di campi per ogni riga
`current_field`: campo corrente

Metodi Principali

`fetch_assoc()`: ottiene riga come array associativo
`fetch_row()`: ottiene riga come array numerico
`fetch_array()`: ottiene riga come array misto
`fetch_all()`: ottiene tutte le righe

Metodi Principali di mysqli_result

fetch_assoc()	Restituisce la riga corrente come array associativo
fetch_row()	Restituisce la riga corrente come array numerico
fetch_array()	Restituisce la riga corrente come array misto (associativo e numerico)

Comportamento

Tutti questi metodi spostano automaticamente l'indicatore interno alla riga successiva dopo ogni chiamata

fetch_assoc() - Array Associativo

Descrizione

Restituisce un array associativo dove le chiavi sono i nomi dei campi

```php
while($row = $result->fetch_assoc()) {
    echo "ID: " . $row["id"];
    echo " - Nome: " . $row["nome"];
    echo " - Cognome: " . $row["cognome"];
}
```

```
["id"] => 1
["nome"] => "Mario"
["cognome"] => "Rossi"
```

https://www.w3schools.com/php/func_mysqli_fetch_assoc.asp

Esempio Completo fetch_assoc()

```php
<?php
$sql = "SELECT id, nome, cognome FROM studenti";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // Output dei dati di ogni riga
    while($row = $result->fetch_assoc()) {
        echo "ID: " . $row["id"];
        echo " - Nome: " . $row["nome"];
        echo " - Cognome: " . $row["cognome"];
        echo "<br>";
    }
} else {
    echo "0 risultati";
}
?>
```

fetch_row() - Array Numerico

Descrizione

Restituisce un array numerico dove gli indici sono posizionali (0, 1, 2, ...)

```php
while($row = $result->fetch_row()) {
    echo "ID: " . $row[0];
    echo " - Nome: " . $row[1];
    echo " - Cognome: " . $row[2];
}
```

```
[0] => 1
[1] => "Mario"
[2] => "Rossi"
```

https://www.w3schools.com/php/func_mysqli_fetch_row.asp

Esempio Completo fetch_row()

```php
<?php
$sql = "SELECT id, nome, cognome FROM studenti";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // Output dei dati di ogni riga
    while($row = $result->fetch_row()) {
        echo "ID: " . $row[0];
        echo " - Nome: " . $row[1];
        echo " - Cognome: " . $row[2];
        echo "<br>";
    }
} else {
    echo "0 risultati";
}
?>
```

Recuperare Tutte le Righe - fetch_assoc()

```php
<?php
$sql = "SELECT id, nome, cognome FROM studenti";
$result = $conn->query($sql);

// Array per memorizzare tutti i risultati
$students = array();

while($row = $result->fetch_assoc()) {
    $students[] = $row;
}

// Utilizzo dell'array
foreach($students as $student) {
    echo $student["nome"] . " " . $student["cognome"];
    echo "<br>";
}
?>
```

Recuperare Tutte le Righe - fetch_row()

```php
<?php
$sql = "SELECT id, nome, cognome FROM studenti";
$result = $conn->query($sql);

// Array per memorizzare tutti i risultati
$students = array();

while($row = $result->fetch_row()) {
    $students[] = $row;
}

// Utilizzo dell'array
foreach($students as $student) {
    echo $student[1] . " " . $student[2];
    echo "<br>";
}
?>
```

Il Metodo fetch_all()

Descrizione

Il metodo `fetch_all()` restituisce TUTTE le righe del risultato in un unico array bidimensionale

Parametri:

resulttype (opzionale) Specifica il tipo di array:

- `MYSQLI_ASSOC` - Array associativo (default)
- `MYSQLI_NUM` - Array numerico
- `MYSQLI_BOTH` - Entrambi i tipi

Vantaggio

Operazione più veloce rispetto al ciclo while con fetch_assoc/fetch_row

Esempio fetch_all() - Base

```php
<?php
$sql = "SELECT id, nome, cognome FROM studenti";
$result = $conn->query($sql);

// Ottiene tutte le righe in un colpo solo
$students = $result->fetch_all();

// Visualizzazione
foreach($students as $student) {
    print_r($student);
    echo "<br>";
}
?>
```

fetch_all() - Utilizzo Pratico

```php
<?php
$sql = "SELECT id, nome, cognome FROM studenti";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    $all_data = $result->fetch_all(MYSQLI_ASSOC);

    echo "<table border='1'>";
    echo "<tr><th>ID</th><th>Nome</th><th>Cognome</th></tr>";

    foreach($all_data as $row) {
        echo "<tr>";
        echo "<td>" . $row["id"] . "</td>";
        echo "<td>" . $row["nome"] . "</td>";
        echo "<td>" . $row["cognome"] . "</td>";
        echo "</tr>";
    }
    echo "</table>";
}
?>
```

fetch_all(MYSQLI_ASSOC)

Array Bidimensionale

```
[0] => ["id"=>1, "nome"=>"Mario",
"cognome"=>"Rossi"]
```

```
[1] => ["id"=>2, "nome"=>"Luigi",
"cognome"=>"Verdi"]
```

```
[2] => ["id"=>3, "nome"=>"Anna",
"cognome"=>"Bianchi"]
```

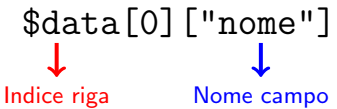
fetch_all(MYSQLI_ASSOC) - Caratteristiche

Struttura

Restituisce un array bidimensionale dove:

- Il primo livello ha indici numerici (0, 1, 2, ...)
- Il secondo livello ha chiavi associative (nomi dei campi)

Accesso ai dati:



fetch_all(MYSQLI_NUM)

Array Bidimensionale Numerico

```
[0] => [0=>1, 1=>"Mario", 2=>"Rossi"]
[1] => [0=>2, 1=>"Luigi", 2=>"Verdi"]
[2] => [0=>3, 1=>"Anna", 2=>"Bianchi"]
```

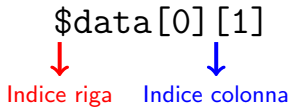
fetch_all(MYSQLI_NUM) - Caratteristiche

Struttura

Restituisce un array bidimensionale dove:

- ENTRAMBI i livelli hanno indici numerici
- Accesso più veloce ma meno leggibile

Accesso ai dati:



Il Metodo close()

Descrizione

Il metodo `close()` chiude la connessione al database MySQL

Parametri:

- Nessun parametro richiesto

Valori Restituiti:

- `true` in caso di successo
- `false` in caso di errore

```php
// Chiusura della connessione
$conn->close();
```

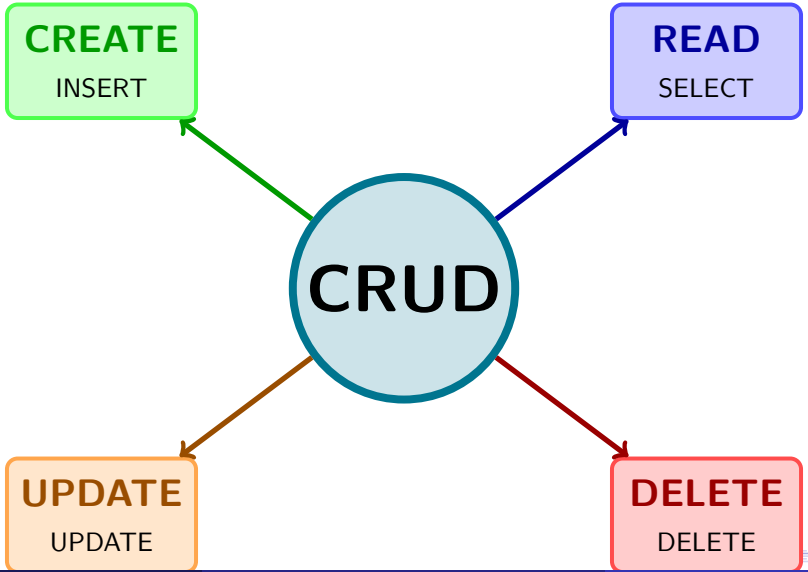
La Proprietà num_rows

Descrizione

Rappresenta il numero di righe nel risultato di una query SELECT

```php
$sql = "SELECT * FROM studenti";
$result = $conn->query($sql);

echo "Numero di record trovati: " . $result->num_rows;

if ($result->num_rows > 0) {
    // Processa i risultati
    while($row = $result->fetch_assoc()) {
        echo $row["nome"];
    }
} else {
    echo "Nessun risultato trovato";
}
```

Operazioni CRUD



Inserimento Record - INSERT

Sintassi SQL

```sql
INSERT INTO nome_tabella (campo1, campo2, campo3)
VALUES (valore1, valore2, valore3);
```

Esempio in PHP:

```php
$nome = "Mario";
$cognome = "Rossi";
$email = "mario.rossi@email.com";

$sql = "INSERT INTO studenti (nome, cognome, email)
        VALUES ('$nome', '$cognome', '$email')";

if ($conn->query($sql) === TRUE) {
    echo "Nuovo record inserito con successo";
} else {
    echo "Errore: " . $conn->error;
}
```

Aggiornamento Dati - UPDATE

Sintassi SQL

```sql
UPDATE nome_tabella
SET campo1 = valore1, campo2 = valore2
WHERE condizione;
```

Esempio in PHP:

```php
$id = 5;
$nuovo_email = "nuova.email@domain.com";

$sql = "UPDATE studenti
        SET email = '$nuovo_email'
        WHERE id = $id";

if ($conn->query($sql) === TRUE) {
    echo "Record aggiornato con successo";
} else {
    echo "Errore: " . $conn->error;
}
```

Cancellazione Record - DELETE

Sintassi SQL

```sql
DELETE FROM nome_tabella WHERE condizione;
```

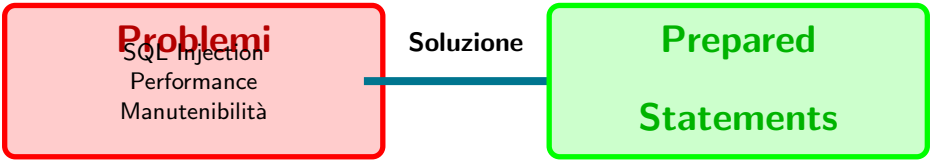
Esempio in PHP:

```php
$id_da_cancellare = 5;

$sql = "DELETE FROM studenti WHERE id = $id_da_cancellare";

if ($conn->query($sql) === TRUE) {
    echo "Record cancellato con successo";
} else {
    echo "Errore: " . $conn->error;
}
```

ATTENZIONE!

Fare sempre attenzione alla clausola WHERE per evitare cancellazioni accidentali

Ottimizzazioni e Sicurezza

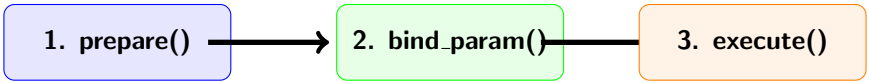


Metodi prepare() e bind_param()

Introduzione

L'utilizzo di `prepare()` e `bind_param()` rappresenta una best practice per:

- Sicurezza (prevenzione SQL injection)
- Performance (ottimizzazione delle query)
- Manutenibilità del codice



Prevenzione SQL Injection

Parametri Disassociati dalla Query

Con `prepare()` e `bind_param()`, i dati vengono separati dalle istruzioni SQL

Query Non Sicura

```
SELECT * FROM users WHERE name = '$name'
```

Vulnerabile a SQL injection!

Migliore

Query Sicura con Prepared Statement

```
SELECT * FROM users WHERE name = ?
```

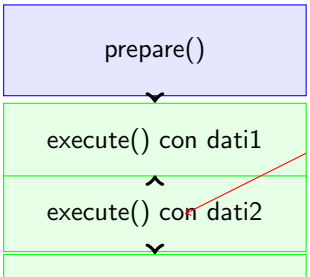
Parametri trattati separatamente

Prestazioni Ottimizzate

Caching del Piano di Esecuzione

La preparazione di una query consente al database di:

- Analizzare la query una sola volta
- Ottimizzare il piano di esecuzione
- Memorizzare in cache il piano
- Riutilizzare lo stesso piano con parametri diversi

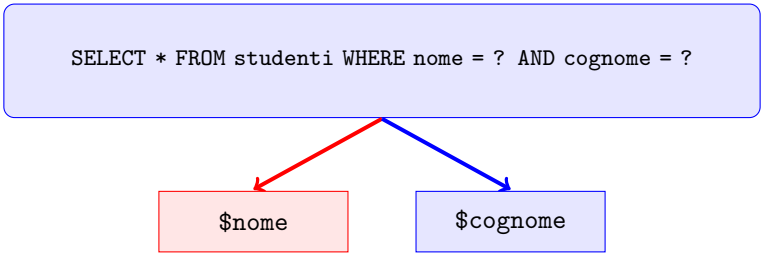


Leggibilità del Codice

Separazione Struttura e Dati

L'utilizzo di segnaposto rende il codice più:

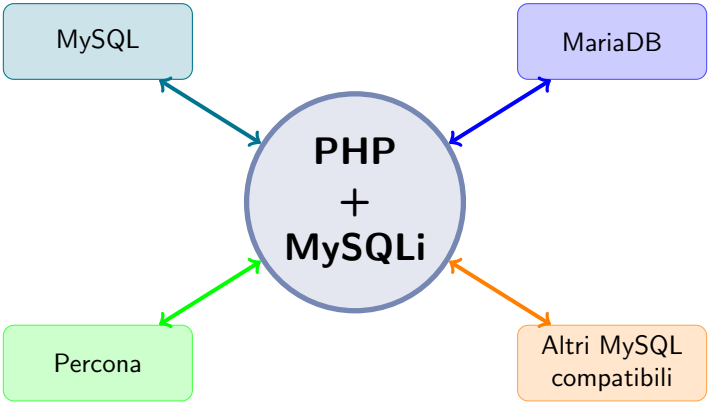
- Leggibile
- Manutenibile
- Comprensibile



Tipi di Dati

`"s"`	String (stringa di testo)
`"i"`	Integer (numero intero)
`"d"`	Double (numero decimale)
`"b"`	Blob (dati binari)

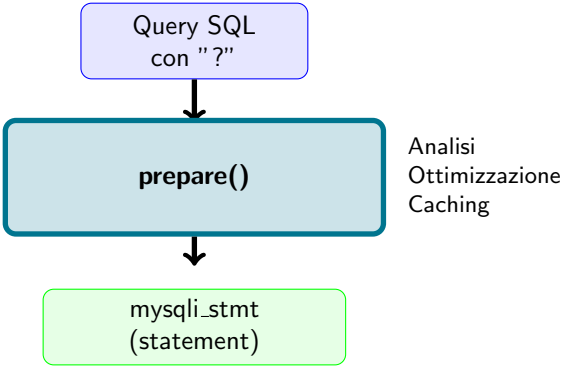
Compatibilità con Database Diversi



Il Metodo prepare()

Descrizione

Il metodo `prepare()` prepara una query SQL per l'esecuzione, creando un piano di esecuzione ottimizzato



Metodo prepare() - Parametri e Ritorno

Parametri

Query (string) La query SQL da preparare, contenente segnaposto "?" per i parametri

Connessione L'oggetto mysqli di connessione al database

Valori di Ritorno

Restituisce un oggetto di tipo `mysqli_stmt` (statement) che rappresenta la query preparata

NOTA

Il segnaposto "?" viene successivamente sostituito con i valori reali tramite `bind_param()`

Esempio Metodo prepare()

```php
<?php
// Query con segnaposto
$sql = "SELECT id, nome, cognome FROM studenti
        WHERE cognome = ? AND anno = ?";

// Preparazione della query
$stmt = $conn->prepare($sql);

if ($stmt === false) {
    die("Errore nella preparazione: " . $conn->error);
}

A questo punto la query    pronta per il binding dei parametri
 e l'esecuzione
?>
```

mysqli_stmt (Statement)

`bind_param()`: Associa parametri ai segnaposto

`execute()`: Esegue la query preparata

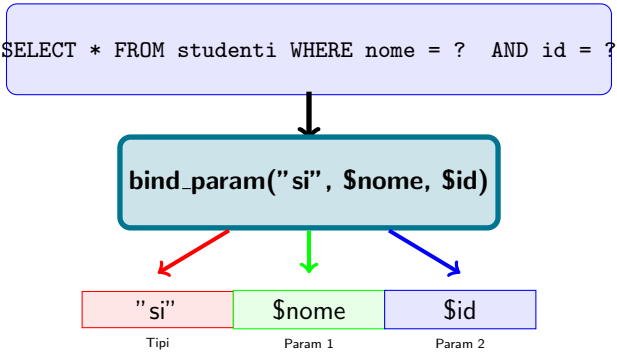
`get_result()`: Ottiene il risultato

`close()`: Chiude lo statement

Il Metodo bind_param()

Descrizione

Il metodo `bind_param()` sostituisce i segnaposto "?" con i valori reali



Metodo bind_param() - Parametri

Parametri

Tipi di dati (string) Una stringa che specifica i tipi:

- "s" - String (stringa)
- "i" - Integer (intero)
- "d" - Double (decimale)
- "b" - Blob (binario)

Parametri (variabili) Le variabili da associare ai segnaposto, nell'ordine

IMPORTANTE

Il numero di caratteri nella stringa dei tipi DEVE corrispondere al numero di parametri

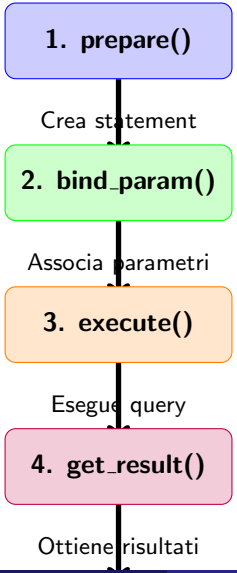
Esempio Completo Prepared Statement

```php
<?php
// 1. Preparazione
$sql = "INSERT INTO studenti (nome, cognome, anno)
        VALUES (?, ?, ?)";
$stmt = $conn->prepare($sql);

// 2. Binding dei parametri
$nome = "Mario";
$cognome = "Rossi";
$anno = 2024;
$stmt->bind_param("ssi", $nome, $cognome, $anno);

// 3. Esecuzione
if ($stmt->execute()) {
    echo "Record inserito con successo";
} else {
    echo "Errore: " . $stmt->error;
}

// 4. Chiusura statement
$stmt->close();
?>
```

Prepared Statement con SELECT

```php
<?php
// Preparazione SELECT
$sql = "SELECT nome, cognome, email FROM studenti
        WHERE anno = ?";
$stmt = $conn->prepare($sql);

// Binding parametro
$anno = 2024;
$stmt->bind_param("i", $anno);

// Esecuzione
$stmt->execute();

// Ottenere i risultati
$result = $stmt->get_result();

while($row = $result->fetch_assoc()) {
    echo $row["nome"] . " " . $row["cognome"];
    echo "<br>";
}

$stmt->close();
```

Schema Completo Prepared Statement

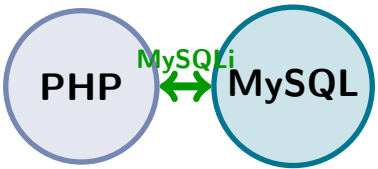


Vantaggi:
- ✓ Sicurezza
- ✓ Performance
- ✓ Riutilizzo
- ✓ Type safety

Riepilogo Finale

Abbiamo studiato:

1. Il DBMS MySQL e phpMyAdmin
2. La classe MySQLi (OOP e procedurale)
3. Connessione e gestione errori
4. Esecuzione query e mysqli_result
5. Metodi fetch (assoc, row, all)
6. Operazioni CRUD
7. Prepared Statements per sicurezza e performance



Grazie per l'Attenzione!

Fedeli Massimo
IIS E.Fermi - Sacconi - Ceci
2024/25