

Programmazione Web con PHP e SQL

Introduzione a SQL

prof. Leonardo Essam Dei Rossi

ITT "M. Buonarroti" - Trento (TN)

Anno scolastico 2025/2026

1 Introduzione

- Gestione del server MySQL
- Schema di funzionamento

2 L'estensione MySQLi

3 Connessione al database

- La classe mysqli
- Il costruttore della classe mysqli
- Gestione degli errori

4 Query con PHP

- Esecuzione di una query

1 Introduzione

- Gestione del server MySQL
- Schema di funzionamento

2 L'estensione MySQLi

3 Connessione al database

- La classe mysqli
- Il costruttore della classe mysqli
- Gestione degli errori

4 Query con PHP

- Esecuzione di una query

Definizione 12.1: Linguaggio SQL

SQL è un linguaggio dichiarativo ad alto livello, standardizzato, per l'interazione con basi di dati relazionali, che permette la definizione delle strutture dati e l'accesso ai dati secondo il modello relazionale.

Domande:

- 1 Quale sotto-insieme di SQL permette la definizione di strutture dati?
- 2 Quale sotto-insieme di SQL permette l'accesso dai dati?

Definizione 12.1: Linguaggio SQL

SQL è un linguaggio dichiarativo ad alto livello, standardizzato, per l'interazione con basi di dati relazionali, che permette la definizione delle strutture dati e l'accesso ai dati secondo il modello relazionale.

Domande:

- 1 Quale sotto-insieme di SQL permette la definizione di strutture dati?
- 2 Quale sotto-insieme di SQL permette l'accesso dai dati?

Definizione 12.1: Linguaggio SQL

SQL è un linguaggio dichiarativo ad alto livello, standardizzato, per l'interazione con basi di dati relazionali, che permette la definizione delle strutture dati e l'accesso ai dati secondo il modello relazionale.

Domande:

- 1 Quale sotto-insieme di SQL permette la definizione di strutture dati?
- 2 Quale sotto-insieme di SQL permette l'accesso dai dati?

Gestione del server MySQL (1)

All'interno di XAMPP è integrato un server MySQL¹ e un'interfaccia Web per una gestione semplice dei vari database/tabelle.

Dalla console di XAMPP avviare sia Apache che MySQL:

Service	Module	PID(s)	Port(s)	Actions			
☐	Apache	10492 6896	80, 443	Stop	Admin	Config	Logs
☐	MySQL	23960	3306	Stop	Admin	Config	Logs

Figura 1.1: Server Apache e MySQL in esecuzione su XAMPP.

¹Server: 10.4.32-MariaDB [UTF-8 Unicode (utf8mb4)]

Gestione del server MySQL (2)

Per accedere all'interfaccia Web di phpMyAdmin è sufficiente accedere al link:

`http://localhost/phpmyadmin/`

Osservare che phpMyAdmin *sembrerebbe* una sotto-cartella del nostro server Web, ma se si osserva attentamente il contenuto della cartella `htdocs` si può notare che la cartella `/phpmyadmin` non è presente.

Questo perché, onde evitare possibili eliminazioni accidentali, la cartella è stata spostata in un posto più sicuro e non direttamente accessibile; per poi essere resa disponibile all'utente tramite un `Alias`.

Gestione del server MySQL (2)

Per accedere all'interfaccia Web di phpMyAdmin è sufficiente accedere al link:

`http://localhost/phpmyadmin/`

Osservare che phpMyAdmin *sembrerebbe* una sotto-cartella del nostro server Web, ma se si osserva attentamente il contenuto della cartella `htdocs` si può notare che la cartella `/phpmyadmin` non è presente.

Questo perché, onde evitare possibili eliminazioni accidentali, la cartella è stata spostata in un posto più sicuro e non direttamente accessibile; per poi essere resa disponibile all'utente tramite un *Alias*.

Per accedere all'interfaccia Web di phpMyAdmin è sufficiente accedere al link:

`http://localhost/phpmyadmin/`

Osservare che phpMyAdmin *sembrerebbe* una sotto-cartella del nostro server Web, ma se si osserva attentamente il contenuto della cartella `htdocs` si può notare che la cartella `/phpmyadmin` non è presente.

Questo perché, onde evitare possibili eliminazioni accidentali, la cartella è stata spostata in un posto più sicuro e non direttamente accessibile; per poi essere resa disponibile all'utente tramite un `Alias`.

Gestione del server MySQL (3)

Ad esempio (solo per completezza):

```
Alias /5inb "D:/2025-2026/5INB/GPOI/ProgettiStudenti"
```

```
<Directory "D:/2025-2026/5INB/GPOI/ProgettiStudenti">
```

```
Options Indexes FollowSymLinks
```

```
AllowOverride All
```

```
Require all granted
```

```
</Directory>
```

In questo modo la cartella D:/2025-2026/5INB/GPOI/ProgettiStudenti verrà resa disponibile tramite il server Web ma senza che la cartella si trovi all'interno di htdocs.

Schema di funzionamento

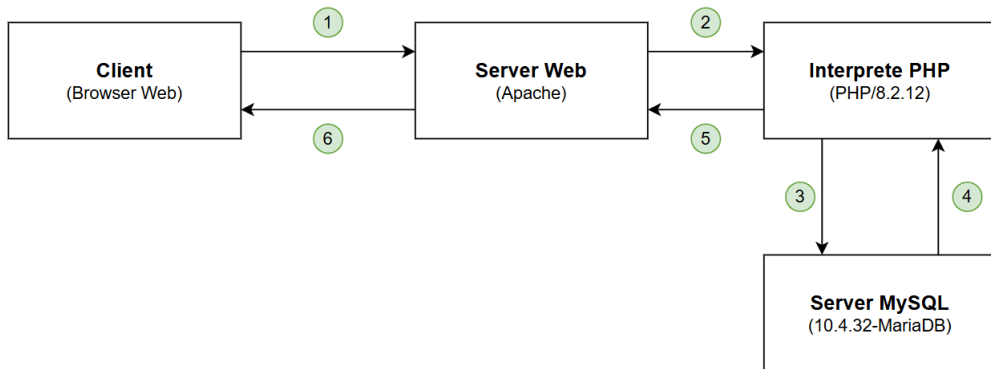


Figura 1.2: Schema di funzionamento.

1 Introduzione

- Gestione del server MySQL
- Schema di funzionamento

2 L'estensione MySQLi

3 Connessione al database

- La classe mysqli
- Il costruttore della classe mysqli
- Gestione degli errori

4 Query con PHP

- Esecuzione di una query

L'estensione MySQLi (1)

MySQLi, acronimo di MySQL Improved, è l'estensione di PHP progettata per consentire l'accesso ai database MySQL e MariaDB in modo moderno, sicuro ed efficiente.

È stata introdotta per superare i limiti della vecchia estensione `mysql`, oggi completamente rimossa, e rappresenta uno degli strumenti principali per lo sviluppo di applicazioni web dinamiche in PHP.

L'estensione MySQLi (2)

L'obiettivo principale di MySQLi è fornire un'interfaccia più robusta e sicura rispetto alle soluzioni precedenti.

L'estensione supporta nativamente i *prepared statement*, che consentono di separare la struttura delle query dai dati inseriti dall'utente, riducendo drasticamente il rischio di *SQL Injection* (*).

MySQLi supporta inoltre transazioni, gestione avanzata degli errori e comunicazione efficiente con il motore MySQL, sfruttando le funzionalità introdotte dalle versioni più recenti del database.

(*) Approfondimento: [1]

MySQLi può essere utilizzata secondo due stili di programmazione.

Lo stile procedurale utilizza funzioni globali ed è più vicino alla sintassi del C, risultando spesso più immediato per chi proviene da una programmazione tradizionale.

Lo stile orientato agli oggetti, invece, incapsula connessione e operazioni in oggetti PHP, risultando più leggibile, modulare e in linea con il PHP moderno.

Dal punto di vista funzionale le due modalità sono equivalenti: la scelta riguarda principalmente lo stile di progettazione del codice.

MySQLi può essere utilizzata secondo due stili di programmazione.

Lo stile procedurale utilizza funzioni globali ed è più vicino alla sintassi del C, risultando spesso più immediato per chi proviene da una programmazione tradizionale.

Lo stile orientato agli oggetti, invece, incapsula connessione e operazioni in oggetti PHP, risultando più leggibile, modulare e in linea con il PHP moderno.

Dal punto di vista funzionale le due modalità sono equivalenti: la scelta riguarda principalmente lo stile di progettazione del codice.

MySQLi può essere utilizzata secondo due stili di programmazione.

Lo stile procedurale utilizza funzioni globali ed è più vicino alla sintassi del C, risultando spesso più immediato per chi proviene da una programmazione tradizionale.

Lo stile orientato agli oggetti, invece, incapsula connessione e operazioni in oggetti PHP, risultando più leggibile, modulare e in linea con il PHP moderno.

Dal punto di vista funzionale le due modalità sono equivalenti: la scelta riguarda principalmente lo stile di progettazione del codice.

MySQLi può essere utilizzata secondo due stili di programmazione.

Lo stile procedurale utilizza funzioni globali ed è più vicino alla sintassi del C, risultando spesso più immediato per chi proviene da una programmazione tradizionale.

Lo stile orientato agli oggetti, invece, incapsula connessione e operazioni in oggetti PHP, risultando più leggibile, modulare e in linea con il PHP moderno.

Dal punto di vista funzionale le due modalità sono equivalenti: la scelta riguarda principalmente lo stile di progettazione del codice.

1 Introduzione

- Gestione del server MySQL
- Schema di funzionamento

2 L'estensione MySQLi

3 Connessione al database

- La classe mysqli
- Il costruttore della classe mysqli
- Gestione degli errori

4 Query con PHP

- Esecuzione di una query

La classe mysqli

La connessione al database MySQL avviene tramite la creazione di una nuova istanza della classe `mysqli`.

Esempio 12.1: Istanza della classe `mysqli`

```
<?php
$connessione = new mysqli(
    "localhost",           // Host
    "root",                // User
    "",                   // Password
    "celin"                // Database
);
?>
```

Il costruttore della classe `mysqli` (1)

Il costruttore della classe `mysqli` presenta diversi argomenti:

- 1 L'indirizzo del server (`hostname`);
- 2 L'utente del server (`user`);
- 3 La password dell'utente (`password`);
- 4 Il nome del database (`database`).

Il costruttore della classe mysqli (2)

L'indirizzo del server (hostname)

Può essere un nome host o un indirizzo IP, quando si passa `null`, il valore viene recuperato da `mysqli.default_host`.

Quando possibile, verranno utilizzate le pipe^a al posto del protocollo TCP/IP.

Il protocollo TCP/IP viene utilizzato se vengono forniti insieme un nome host e un numero di porta, ad esempio `localhost:3308`.

^aIn Windows, socket interni al sistema (come i socket locali su Linux).

Approfondimento: [2]

Il costruttore della classe mysqli (2)

L'indirizzo del server (hostname)

Può essere un nome host o un indirizzo IP, quando si passa `null`, il valore viene recuperato da `mysqli.default_host`.

Quando possibile, verranno utilizzate le pipe^a al posto del protocollo TCP/IP.

Il protocollo TCP/IP viene utilizzato se vengono forniti insieme un nome host e un numero di porta, ad esempio `localhost:3308`.

^aIn Windows, socket interni al sistema (come i socket locali su Linux).

Approfondimento: [2]

Il costruttore della classe mysqli (3)

L'utente del server (user)

Il nome utente MySQL oppure `null` per assumere il nome utente in base all'opzione `mysqli.default_user`.

La password dell'utente (password)

La password MySQL o `null` per assumere la password in base all'opzione `mysqli.default_pw`.

Il nome del database (database)

Il database predefinito da utilizzare quando si eseguono query o `null`.

Il costruttore della classe mysqli (3)

L'utente del server (user)

Il nome utente MySQL oppure `null` per assumere il nome utente in base all'opzione `mysqli.default_user`.

La password dell'utente (password)

La password MySQL o `null` per assumere la password in base all'opzione `mysqli.default_pw`.

Il nome del database (database)

Il database predefinito da utilizzare quando si eseguono query o `null`.

Il costruttore della classe mysqli (3)

L'utente del server (user)

Il nome utente MySQL oppure `null` per assumere il nome utente in base all'opzione `mysqli.default_user`.

La password dell'utente (password)

La password MySQL o `null` per assumere la password in base all'opzione `mysqli.default_pw`.

Il nome del database (database)

Il database predefinito da utilizzare quando si eseguono query o `null`.

Il costruttore della classe `mysqli` in ogni caso restituirà un oggetto di tipo `mysqli`, anche in caso di errore all'atto della connessione.

Viene lasciata quindi allo sviluppatore la responsabilità di gestire l'errore all'atto della connessione.

Il costruttore della classe `mysqli` in ogni caso restituirà un oggetto di tipo `mysqli`, anche in caso di errore all'atto della connessione.

Viene lasciata quindi allo sviluppatore la responsabilità di gestire l'errore all'atto della connessione.

Esempio 12.2: Errore di connessione

```
<?php
$mysqli = new mysqli("localhost", "root", "", "celin");
if ($mysqli->connect_errno) {
    echo("Errore: " . $mysqli->connect_error);
}
?>
```

Alcune osservazioni:

- `$mysqli->connect_errno`: codice di errore numerico dell'ultimo errore che si è verificato;
- `$mysqli->connect_error`: descrizione testuale dell'ultimo errore che si è verificato.

Esempio 12.2: Errore di connessione

```
<?php
$mysqli = new mysqli("localhost", "root", "", "celin");
if ($mysqli->connect_errno) {
    echo("Errore: " . $mysqli->connect_error);
}
?>
```

Alcune osservazioni:

- `$mysqli->connect_errno`: codice di errore numerico dell'ultimo errore che si è verificato;
- `$mysqli->connect_error`: descrizione testuale dell'ultimo errore che si è verificato.

Esempio 12.2: Errore di connessione

```
<?php
$mysqli = new mysqli("localhost", "root", "", "celin");
if ($mysqli->connect_errno) {
    echo("Errore: " . $mysqli->connect_error);
}
?>
```

Alcune osservazioni:

- `$mysqli->connect_errno`: codice di errore numerico dell'ultimo errore che si è verificato;
- `$mysqli->connect_error`: descrizione testuale dell'ultimo errore che si è verificato.

1 Introduzione

- Gestione del server MySQL
- Schema di funzionamento

2 L'estensione MySQLi

3 Connessione al database

- La classe mysqli
- Il costruttore della classe mysqli
- Gestione degli errori

4 Query con PHP

- Esecuzione di una query

Esecuzione di una query (1)

Esempio 12.3: Esecuzione di una query

```
<?php
$mysqli = new mysqli("localhost", "root", "", "celin");
$result = $mysqli->query(
    "SELECT * FROM Corso WHERE Codice = 145008"
);

$row = $result->fetch_assoc();
?>
```

Il metodo `query(...)` dell'oggetto `mysqli` ammette e richiede come parametro una stringa che rappresenta la query da inviare al server MySQL.

Restituisce a sua volta un oggetto di tipo `mysqli_result`.

Esecuzione di una query (1)

Esempio 12.3: Esecuzione di una query

```
<?php
$mysqli = new mysqli("localhost", "root", "", "celin");
$result = $mysqli->query(
    "SELECT * FROM Corso WHERE Codice = 145008"
);

$row = $result->fetch_assoc();
?>
```

Il metodo `query([...])` dell'oggetto `mysqli` ammette e richiede come parametro una stringa che rappresenta la query da inviare al server MySQL.

Restituisce a sua volta un oggetto di tipo `mysqli_result`.

Esecuzione di una query (2)

Per accedere agli elementi, vale il metodo:

```
$row = $result->fetch_assoc();
```

In questo modo si va a leggere la prima tupla ricavata dalla query SQL.

Per iterare tutte le tuple ricavate, vale:

```
while ($row = $result->fetch_assoc()) {  
    // ...  
}
```

Approfondimento (1): [3]

Approfondimento (2): [4]

Esecuzione di una query (2)

Per accedere agli elementi, vale il metodo:

```
$row = $result->fetch_assoc();
```

In questo modo si va a leggere la prima tupla ricavata dalla query SQL.

Per iterare tutte le tuple ricavate, vale:

```
while ($row = $result->fetch_assoc()) {  
    // ...  
}
```

Approfondimento (1): [3]

Approfondimento (2): [4]

Esecuzione di una query (2)

Per accedere agli elementi, vale il metodo:

```
$row = $result->fetch_assoc();
```

In questo modo si va a leggere la prima tupla ricavata dalla query SQL.

Per iterare tutte le tuple ricavate, vale:

```
while ($row = $result->fetch_assoc()) {  
    // ...  
}
```

Approfondimento (1): [3]

Approfondimento (2): [4]

- [1] Cloudflare, *Cos'è l'SQL injection?* [Online; accessed 25-February-2026], 2026. indirizzo: <https://www.cloudflare.com/it-it/learning/security/threats/sql-injection/>.
- [2] MySQL developers, *Local Socket/Pipe Connection Method*, [Online; accessed 25-February-2026], 2026. indirizzo: <https://dev.mysql.com/doc/workbench/en/wb-mysql-connections-methods-socket.html>.
- [3] php.net authors, *The mysqli class*, [Online; accessed 25-February-2026], 2026. indirizzo: <https://www.php.net/manual/en/class.mysqli.php>.
- [4] php.net authors, *The mysqli_result class*, [Online; accessed 25-February-2026], 2026. indirizzo: <https://www.php.net/manual/en/class.mysqli-result.php>.