

# Videocorso PHP



## Le basi di dati

richiami di SQL e collegamento a PHP

*Alessandro Flora*

# Le basi di dati

## Come sarà impostata questa lezione

- All'interno delle slide ci saranno alcuni riferimenti utili per il successivo ripasso
- In particolare nelle slide sono riportati i concetti di base e la sintassi
- La seconda parte sarà un esercizio svolto; è vivamente consigliato provare a ragionare insieme creando uno script per provare in prima persona



`index.php`

# Le basi di dati

## Cos'è una base di dati

- È una collezione di dati organizzati, in particolare ci occuperemo del modello relazionale
- È utile nelle applicazioni web perché consente di immagazzinare i dati degli utenti (o anche di altro tipo) e accedervi senza particolari vincoli
- Rende il nostro codice quanto più flessibile possibile in molte circostanze



# Le basi di dati

## Cos'è una base di dati relazionale

- È un modello matematico che memorizza i dati sotto forma di valori atomici posti in relazione tra di essi
- È composta da tabelle di dimensione fissa in cui ogni colonna rappresenta un dato ed è identificata da una chiave univoca detta primaria
- Le tabelle possono essere messe in relazione mediante le chiavi esterne
- È un modello molto efficiente per quanto riguarda la ricerca dei valori
- La loro progettazione segue delle regole di base

## Dal modello ER alla base di dati relazionale

- Ogni entità del modello relazionale diventa il modello per creare una tabella le cui colonne sono le informazioni salvate
- Identifichiamo il/i dati univoci di ogni entità: questi saranno la chiave primaria
- Le relazioni 1:N o 1:1 pongono la chiave esterna come colonna della tabella (o delle tabelle che hanno cardinalità 1)
- Le relazioni N:N necessitano di una tabella esterna per memorizzare le associazioni; tale tabella presenterà tra le colonne le due chiavi esterne
- Può essere utile impostare il modello logico come passaggio intermedio

# Le basi di dati

## Il linguaggio SQL

Structured Query Language

È un linguaggio standardizzato per effettuare delle operazioni sulle basi di dati relazionali, in particolare lo possiamo usare per:

- creare o modificare i database e le tabelle di essi
- inserire, modificare o eliminare i dati inseriti nelle tabelle
- consultare i dati memorizzati
- creare e gestire strumenti di accesso ai dati (gestire i permessi, gli utenti, ...)

Ogni operazione tra le precedenti su una base di dati è detta **query** ed è molto vicino per rigidità della sintassi a un linguaggio di programmazione.

# Le basi di dati

## IL RDBMS MariaDB

- È il *Relational Database Management System* (componente software che gestisce la base di dati) che useremo
- È un *fork* di MySQL del suo sviluppatore originario (sono entrambi open-source ma MariaDB è decisamente più community-oriented)
- Lo useremo insieme a un motore di memorizzazione (InnoDB) per getire i dati e le query SQL



# Le basi di dati

## phpMyAdmin

- È un software scritto in PHP che utilizzeremo come intermediario tra noi e MariaDB per poter creare e modificare le basi di dati in maniera grafica
- Sui computer del laboratorio lo potete raggiungere all'indirizzo **`http://localhost:99`** e potete accedere usando le credenziali:

username: **studente**

password: **studente**





## Impostare una query per manipolare i dati

- Seguono una sintassi rigida e possono essere dei seguenti tipi:
  - **SELECT** per leggere uno o più record di una (o più) tabelle della base di dati
  - **INSERT** per inserire un record in una tabella della base di dati
  - **UPDATE** per modificare uno o più campi di uno o più record di una tabella del db
  - **DELETE** per eliminare uno o più record di una tabella della base di dati
  - **DROP** per eliminare completamente una tabella o una base di dati
  - **TRUNCATE** per eliminare completamente i record di una tabella

# Le basi di dati

## La query SELECT

```
SELECT campo1, campo2  
FROM tabella  
WHERE ID= '1';
```

Può anche essere scritta in linea e,  
nel caso in cui non sia seguita da  
altro codice, si può omettere il ;



```
SELECT * FROM tabella WHERE ID= '1'
```

Questa è la notazione che useremo in  
PHP per effettuare le query (in linea  
e senza ;)

tabella

ID	campo1	campo2
1	valore1	valore2
2	valore3	valore4

Inserire un asterisco dopo il **SELECT**  
selezionerà tutti i campi della tabella



```
SELECT *  
FROM tabella;
```

Possiamo omettere la condizione  
**WHERE** quando vogliamo selezionare  
tutti i valori della tabella

# Le basi di dati

## La query SELECT su più tabelle

```
SELECT tabella.campo1, tabella2.campo1
FROM tabella, tabella2
WHERE tabella.ID=tabella2.ID_tab1
AND tabella.ID='2';
```



Per inserire più di una condizione nel **WHERE** usiamo le congiunzioni logiche **AND**, **OR**, **XOR**, **NOT** (analoghi a quelli visti per PHP e JS), gli operatori di confronto restano anch'essi analoghi



Nel **WHERE** dobbiamo indicare anche la relazione tra i record delle due tabelle, in questo caso **ID\_tab1** è la chiave esterna di **tabella2** rispetto a **tabella**

tabella

ID	campo1	campo2
1	valore1	valore2
2	valore3	valore4

tabella2

ID	ID_tab1	campo1
1	2	valore

# Le basi di dati

## La query INSERT

```
INSERT INTO tabella(ID,campo1,campo2)  
VALUES('3','valore5','valore6');
```



È anche possibile omettere il nome dei campi se li stiamo inserendo tutti; possiamo omettere un campo se questo ha un valore predefinito (come NULL) oppure se è soggetto ad auto-increment

tabella

ID	campo1	campo2
1	valore1	valore2
2	valore3	valore4

# Le basi di dati

## La query UPDATE

```
UPDATE tabella  
SET campo1='nuovo_valore', campo2='valore'  
WHERE ID='1';
```



Possiamo modificare quanti campi vogliamo inserendo una virgola tra ogni campo da modificare.  
Se non inseriamo il **WHERE** con la relativa condizione modificheremo l'intera tabella.

tabella		
ID	campo1	campo2
1	valore1	valore2
2	valore3	valore4

## La query DELETE

```
DELETE FROM tabella  
WHERE ID= '1';
```



Se non inseriamo una condizione con **WHERE** verrà eliminato il contenuto dell'intera tabella.  
Senza l'uso delle transazioni (come stiamo facendo in questi esempi) eliminiamo permanentemente le righe affette da questa query ma non resettiamo i contatori dell'auto-increment

tabella		
ID	campo1	campo2
1	valore1	valore2
2	valore3	valore4

## Le query DROP e TRUNCATE

```
TRUNCATE TABLE tabella;  
DROP TABLE tabella;  
DROP DATABASE db;
```



La prima query andrà a svuotare la tabella resettando eventuali auto-increment.

La seconda query rimuoverà la tabella.

La terza query rimuoverà l'intero database

# Connettere il DB a PHP

## La classe mysqli

- È una classe (insieme a PDO) che consente di stabilire una connessione a una base di dati MySQL (o MariaDB) ed effettuare query
- Sono necessari alcuni elementi di programmazione a oggetti per comprendere come si stabilisce la connessione
- Si possono seguire due approcci: uno injection-safe e uno injection-unsafe; noi seguiremo il secondo
- La documentazione PHP relativa alla classe è disponibile all'indirizzo <https://www.php.net/manual/it/book.mysqli.php>



# Connettere il DB a PHP

## Connettersi a un DB

```
$db = new mysqli($server,$nome_utente,$password,$database);
```



Costruiamo un'istanza della classe mysqli

```
$db = new mysqli('localhost','studente','studente','studente');
```

Sintassi della chiamata al costruttore per il setup del laboratorio e dei video di configurazione

# Connettere il DB a PHP

## Preparare una query

```
$query = $db->prepare("SELECT * FROM tabella WHERE ID=?");  
$query->bind_param('i',$id);
```



Il punto interrogativo segnala che andrà inserito un parametro



Il primo parametro è una stringa che definisce il tipo dei parametri e può essere uno dei seguenti:

- **i** per gli interi
- **s** per le stringhe
- **d** per i float/double

Se ci fossero più ? da inseriamo più lettere nella stessa stringa, per esempio 'iisdsi'



I parametri successivi sono le variabili che contengono i valori da sostituire ai punti interrogativi

# Connettere il DB a PHP

## Inoltrare la query e ottenerne i risultati

```
$query->execute();  
$risposta = $query->get_result();
```



Risposta diviene un'istanza della classe `mysqli_result`, nella slide successiva sono presentati alcuni metodi della classe utile per accedere ai risultati

```
$risposta->num_rows;
```

Restituisce come intero il numero di righe della risposta (è un attributo della classe e dunque non ha le tonde finali)

# Connettere il DB a PHP

## Metodi per ottenere i risultati: #1 vettore di vettori associativi

```
$righe = $risposta->fetch_all(MYSQLI_ASSOC);
```

```
$righe = [  
    [  
        'ID' => 1,  
        'campo1' => 'valore1',  
        'campo2' => 'valore2'  
    ],  
    [  
        'ID' => 2,  
        'campo1' => 'valore3',  
        'campo2' => 'valore4'  
    ]  
];
```



È una costante numerica definita nella classe e serve a specificare che i sottovettori (uno per ogni riga della risposta) devono essere associativi

tabella		
ID	campo1	campo2
1	valore1	valore2
2	valore3	valore4

# Connettere il DB a PHP

## Metodi per ottenere i risultati: #2 vettore di vettori classici

```
$righe = $risposta->fetch_all(MYSQLI_ASSOC);
```

```
$righe = [  
    [  
        'ID' => 1,  
        'campo1' => 'valore1',  
        'campo2' => 'valore2'  
    ],  
    [  
        'ID' => 2,  
        'campo1' => 'valore3',  
        'campo2' => 'valore4'  
    ]  
];
```



tabella		
ID	campo1	campo2
1	valore1	valore2
2	valore3	valore4

È una costante numerica definita nella classe e serve a specificare che i sottovettori (uno per ogni riga della risposta) devono essere associativi

# Connettere il DB a PHP

## Metodi per ottenere i risultati: #2 vettore di vettori classici

```
$righe = $risposta->fetch_all(MYSQLI_NUM);
```

```
$righe = [  
    [1, 'valore1', 'valore2'],  
    [2, 'valore3', 'valore4']  
];
```



tabella

ID	campo1	campo2
1	valore1	valore2
2	valore3	valore4

È una costante numerica definita nella classe e serve a specificare che i sottovettori (uno per ogni riga della risposta) devono essere classici.

Possiamo ometterla visto che è il valore di default

# Connettere il DB a PHP

**Metodi per ottenere i risultati: #3 iteriamo sulle righe come vett. class.**

```
while($riga = $risposta->fetch_row()) {  
    // codice  
}
```

```
// alla prima esecuzione  
$riga = [1, 'valore1', 'valore2'];
```

```
// alla seconda esecuzione  
$riga = [2, 'valore3', 'valore4']
```

tabella

ID	campo1	campo2
1	valore1	valore2
2	valore3	valore4

# Connettere il DB a PHP

**Metodi per ottenere i risultati: #4 iteriamo sulle righe come vett. assoc.**

```
while($riga = $risposta->fetch_assoc()) {  
    // codice  
}  
  
// alla prima esecuzione  
$riga = [  
    'ID' => 1,  
    'campo1' => 'valore1',  
    'campo2' => 'valore2'  
];  
  
// alla seconda esecuzione  
$riga = [  
    'ID' => 2,  
    'campo1' => 'valore3',  
    'campo2' => 'valore4'  
];
```

tabella		
ID	campo1	campo2
1	valore1	valore2
2	valore3	valore4



# Connettere il DB a PHP

## Chiudere la connessione al DB

```
$db->close();
```



Al termine dell'uso del database è ottima norma chiudere la connessione mediante il metodo **close()**