IIS PRIMO LEVI TORINO A.S. 2025/26 INFORMATICA CLASSE 5C

Autori: Franco Sicca e chatGPT

[**Basi di Dati, SQL e XAMPP 4**](#_sqfkgeuxo09)

[1. Introduzione ai Database 4](#_s6wp12q9jttz)

[Cos'è un database? 4](#_qd47a3y8fqcu)

[Tipi di database 4](#_1y8v5e1hpqmo)

[2. Il Modello Relazionale 4](#_sbw9exgave7a)

[Concetti base 4](#_fw0b14bgiu9f)

[3. Introduzione a SQL 5](#_j9ezixjpfyyz)

[SQL – Structured Query Language 5](#_amtysb6950kf)

[Tipi di comandi: 5](#_6arxwf96c1iq)

[Sintassi base 5](#_gj38a3fntunp)

[Creare una tabella: 5](#_5m1em4n8kpot)

[Inserire dati: 5](#_dwyl14t6kfz6)

[Visualizzare dati: 6](#_1y2oizlgm84y)

[Filtrare: 6](#_xz5v8z99gjix)

[Modificare un dato: 6](#_muwge9ggique)

[Cancellare un record: 6](#_2gim6o1pdajh)

[4. XAMPP: Ambiente di sviluppo Web 6](#_ot9uws5hr2b2)

[Cos'è XAMPP? 6](#_mw4zdvm1vo4r)

[Perché usarlo? 6](#_y8qicxwfu17p)

[5. Installazione e Primo Avvio di XAMPP 6](#_udu5klekl0ig)

[6. phpMyAdmin 7](#_y9eeodj1fgfq)

[7. Esercizi 7](#_x5rds6u6mnsf)

[Esercizio 1 – Crea il tuo primo database 7](#_ff7ztim0u20l)

[Esercizio 2 – Relazioni 8](#_fx81qznspp4i)

[8. Consolidamento e approfondimenti futuri 8](#_f8x6l5k4vl31)

[9. Progettare un Database: il Modello Concettuale 8](#_3tpu37fksuw)

[Fasi della progettazione di un database 8](#_uodqt7i7p13y)

[10. Tipi di relazione 9](#_nrj2fo4yqxgu)

[Relazione 1:1 (uno a uno) 9](#_67yncnu88uht)

[Definizione: 9](#_doavesa54285)

[Esempio: 9](#_nq0782n409od)

[Schema E-R: 9](#_yanp48bxfnwc)

[Implementazione: 9](#_ct7aeeq4bis2)

[Relazione 1:N (uno a molti) 10](#_5ljnx7888kd0)

[Definizione: 10](#_mged9dw8h8sp)

[Esempio: 10](#_ktiwngv9i4v8)

[Schema E-R: 10](#_4abpmi2d63it)

[Implementazione: 10](#_jfzvqbgxe4l)

[Relazione N:M (molti a molti) 11](#_oun9i798ten)

[Definizione: 11](#_xt7bgw4hbuqi)

[Esempio: 11](#_kphzyt3ng2ky)

[Schema E-R: 11](#_d19y9tva5b17)

[Implementazione: 11](#_atj1l5yl43h3)

[11. Consigli per una buona progettazione concettuale 12](#_lvohmi42s0cm)

[12 UN ESEMPIO: Relazioni tra Docenti e Studenti 12](#_3htdtqabbmzv)

[Contesto scolastico: Istituto Tecnico Industriale 12](#_no0kbu9nc0cr)

[🔹 Relazione 1-a-N tra Docente di Informatica e Studente 12](#_knse3sgl2f9k)

[Situazione: 12](#_aiywmzvwl7zr)

[➤ Tipo di relazione: 12](#_cvhdr5mz8q5p)

[➤ Modello concettuale: 12](#_tlrnnme1on7z)

[➤ Implementazione: 13](#_740ln8ewm5td)

[🔸 Relazione N-a-M tra Docente Generico e Studente 13](#_y470lan95kct)

[Situazione: 13](#_8r0326dop1u6)

[➤ Tipo di relazione: 13](#_q43dv6kg0f21)

[➤ Modello concettuale: 13](#_s2fu4v97k8nz)

[Nella relazione possiamo aggiungere attributi come: 14](#_adrqfvg2tdcj)

[➤ Implementazione (con tabella intermedia): 14](#_2cwi82org6z4)

[🔁 Riepilogo 14](#_mc5mhcg6idjk)

[13 UN ESEMPIO DI SIMULAZIONE DELLA PROVA DELLA MATURITA’ 15](#_7w3neh2w7ew1)

[🎯 Traccia – Sintesi 15](#_ahb7xcntxe17)

[📐 Modello concettuale (ER) 15](#_m90f8garg15s)

[Schema logico e tabelle SQL 16](#_ju26akyevhgz)

[Query SQL richieste 17](#_hl572zwv8kky)

[Spiegazione del database progettato 18](#_dp4vdv99l5e8)

[14 NORMALIZZAZIONE DI UN DATABASE 21](#_9njzvhns5lgp)

[🔹 PRIMA FORMA NORMALE (1NF) 21](#_ba36rbfu4m2c)

[🔸 SECONDA FORMA NORMALE (2NF) 21](#_gi60joild96)

[🔺 TERZA FORMA NORMALE (3NF) 22](#_hih49nhpln9z)

[🧠 Riepilogo visivo 23](#_pc0pyupnz3xq)

[15 SINTASSI SQL – GUIDA PRATICA CON ESEMPI 25](#_jeb53gqoasbz)

[✅ 1. DDL – Data Definition Language 25](#_4zrpj4fj8qe2)

[📌 Esempio: creare le tabelle Studente e Corso 25](#_lvzhab4uwooo)

[➕ Aggiunta colonna alla tabella 26](#_qhpyaocycia)

[❌ Eliminazione tabella 26](#_wonyvsb9vtxl)

[✍️ 2. DML – Data Manipulation Language 26](#_cz1r7itxi26f)

[📌 Inserimento di dati 26](#_fso3deuvq30s)

[🛠️ Modifica di un dato 26](#_1rxqd6rtpmbo)

[🗑️ Cancellazione di un dato 26](#_o6x27nwyzjro)

[🔎 3. DQL – Data Query Language 27](#_ub4c5cwm92mb)

[📌 Esempi: 27](#_s0hcx2yg5iqy)

[🧠 Riepilogo 28](#_q8h8q84p3v4d)

[16 Cos’è XAMPP 28](#_afespq8wzxzq)

[✅ Cosa include XAMPP? 28](#_skhvx8f15nfa)

[🗃️ Cos’è MySQL 28](#_pnfv4z62cp0m)

[🧠 1. MySQL come DBMS (Database Management System) 28](#_d71g46vagzik)

[🔍 Esempio: 29](#_8zrk039u2mpz)

[Come funziona nella pratica con XAMPP 30](#_hvn9ubr839su)

[🧠 Riepilogo 30](#_b54ixa1svip4)

[18 Apache, il Web Server, e l’architettura client-server 30](#_tdqslimmr0g)

[🔥 Cos’è Apache? 30](#_4uomgrjgjeq3)

[🖥️ Dove si trova Apache? 31](#_tnzxtbdj75f1)

[🧭 Architettura Client - Server in sintesi 32](#_415whckwbubj)

[🔁 Esempio reale (produzione) 32](#_emlnug2570d6)

[🧪 Esempio in locale con XAMPP 32](#_cu1u7ekdczaa)

[📌 Conclusione 33](#_e928fys1453e)

# **Basi di Dati, SQL e XAMPP**

## **1. Introduzione ai Database**

### **Cos'è un database?**

Un **database** è un insieme organizzato di dati strutturati, gestito tramite un **DBMS** (Database Management System). Un DBMS permette di:

* Archiviare grandi quantità di dati
* Interrogarli in modo rapido
* Aggiornarli, cancellarli, filtrarli
* Gestire la concorrenza (accessi multipli)
* Garantire sicurezza e integrità

### **Tipi di database**

* **Relazionali** (es. MySQL, PostgreSQL, SQLite)
* **Non relazionali (NoSQL)** (es. MongoDB, Redis)

## **2. Il Modello Relazionale**

### **Concetti base**

* **Tabella**: rappresenta un'entità (es. Studenti, Corsi)
* **Record**: riga di una tabella (un'istanza dell'entità)
* **Campo (colonna)**: proprietà dell'entità
* **Chiave primaria**: campo che identifica univocamente un record
* **Chiave esterna**: riferimento a una chiave primaria in un'altra tabella

Esempio:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID\_Studente** | **Nome** | **Cognome** | **Classe** |
| 1 | Marco | Rossi | 5A |
| 2 | Laura | Bianchi | 5B |

## **3. Introduzione a SQL**

### **SQL – Structured Query Language**

È il linguaggio standard per interagire con database relazionali.

#### **Tipi di comandi:**

* **DDL (Data Definition Language)** – creazione/modifica tabelle
* **DML (Data Manipulation Language)** – inserimento, modifica, cancellazione dati
* **DQL (Data Query Language)** – interrogazione dati

### **Sintassi base**

#### **Creare una tabella:**

sql

CopiaModifica

CREATE TABLE Studenti (

ID INT PRIMARY KEY,

Nome VARCHAR(50),

Cognome VARCHAR(50),

Classe VARCHAR(5)

);

#### **Inserire dati:**

sql

CopiaModifica

INSERT INTO Studenti (ID, Nome, Cognome, Classe)

VALUES (1, 'Marco', 'Rossi', '5A');

#### **Visualizzare dati:**

SELECT \* FROM Studenti;

#### **Filtrare:**

SELECT \* FROM Studenti WHERE Classe = '5A';

#### **Modificare un dato:**

UPDATE Studenti SET Classe = '5B' WHERE ID = 1;

#### **Cancellare un record:**

DELETE FROM Studenti WHERE ID = 1;

## **4. XAMPP: Ambiente di sviluppo Web**

### **Cos'è XAMPP?**

**XAMPP** è un pacchetto software gratuito che include:

* **X**: Multipiattaforma (Windows, Linux, Mac)
* **A**: Apache (Web server)
* **M**: MySQL (Database server)
* **P**: PHP (linguaggio di programmazione)
* **P**: Perl (linguaggio di programmazione)

### **Perché usarlo?**

Fornisce un ambiente completo per sviluppare applicazioni web in locale, senza bisogno di un server online.

## **5. Installazione e Primo Avvio di XAMPP**

1. **Scarica** XAMPP da<https://www.apachefriends.org>
2. **Installa** seguendo la procedura guidata
3. Avvia **XAMPP Control Panel**
4. Avvia i moduli **Apache** e **MySQL**
5. Visita http://localhost nel browser

## **6. phpMyAdmin**

Strumento incluso in XAMPP per gestire MySQL via interfaccia grafica:

* URL: http://localhost/phpmyadmin
* Permette di:  
  + Creare database
  + Eseguire query SQL
  + Importare/esportare dati
  + Gestire utenti

## **7. Esercizi**

### Esercizio 1 – Crea il tuo primo database

* Crea un database chiamato Scuola
* Crea una tabella Studenti
* Inserisci almeno 3 record
* Interroga i dati con SELECT

### Esercizio 2 – Relazioni

* Aggiungi una tabella Corsi con:  
  + IDCorso, NomeCorso, Classe
* Crea una relazione tra Studenti e Corsi
* Usa JOIN per visualizzare i dati correlati

SELECT Studenti.Nome, Corsi.NomeCorso

FROM Studenti

JOIN Corsi ON Studenti.Classe = Corsi.Classe;

## **8. Consolidamento e approfondimenti futuri**

* CRUD con PHP
* Sicurezza (iniezione SQL)
* Progettazione E-R
* Normalizzazione

## **9. Progettare un Database: il Modello Concettuale**

La progettazione di un database parte da un’**analisi concettuale**: capire “cosa” deve essere rappresentato, prima di pensare a “come” verrà implementato in SQL.

### **Fasi della progettazione di un database**

1. **Analisi dei requisiti**
   * Raccogliere informazioni su ciò che il sistema deve fare.
   * Parlare con gli utenti finali.
   * Identificare entità, attributi e relazioni.
2. **Modello Entità-Relazione (E-R)**
   * Usato per rappresentare graficamente le informazioni raccolte.
   * Include:  
     + **Entità** (sostantivi: Studente, Corso, Docente…)
     + **Attributi** (proprietà: nome, data, voto…)
     + **Relazioni** (verbi: segue, insegna, partecipa…)
3. **Traduzione in modello logico (relazionale)**
   * Conversione delle entità e relazioni in **tabelle SQL**.
   * Definizione di **chiavi primarie** e **chiavi esterne**.

## **10. Tipi di relazione**

### **Relazione 1:1 (uno a uno)**

#### **Definizione:**

A ogni entità A corrisponde **al massimo una** entità B, e viceversa.

#### **Esempio:**

Ogni **persona** ha una **carta d’identità**, e ogni carta è assegnata a una sola persona.

#### **Schema E-R:**

* Entità: Persona, CartaIdentità
* Relazione: possiede (1:1)

#### **Implementazione:**

Può essere fusa in un’unica tabella o separata, con una **chiave esterna unica**.

CREATE TABLE Persona (

ID INT PRIMARY KEY,

Nome VARCHAR(50)

);

CREATE TABLE CartaIdentita (

Numero INT PRIMARY KEY,

IDPersona INT UNIQUE,

FOREIGN KEY (IDPersona) REFERENCES Persona(ID)

);

### **Relazione 1:N (uno a molti)**

#### **Definizione:**

A un'entità A possono corrispondere **più** entità B, ma ogni entità B appartiene **a una sola** A.

#### **Esempio:**

Un **docente di informatica** può insegnare a **molte classi**, ma ogni **classe** ha un solo docente di informatica.

#### **Schema E-R:**

* Entità: Docente, Classe
* Relazione: insegna (1:N)

#### **Implementazione:**

Si inserisce una **chiave esterna** nella tabella "molti".

sql

CopiaModifica

CREATE TABLE DocenteINFORMATICA (

ID INT PRIMARY KEY,

Nome VARCHAR(50)

);

CREATE TABLE Classe (

ID INT PRIMARY KEY,

Sezione VARCHAR(5),

IDDocente INT,

FOREIGN KEY (IDDocente) REFERENCES Docente(ID)

);

### **Relazione N:M (molti a molti)**

#### **Definizione:**

A ogni entità A possono corrispondere **più** entità B, e viceversa.

#### **Esempio:**

Uno **studente** può seguire **più corsi**, e ogni **corso** può essere seguito da più studenti.

#### **Schema E-R:**

* Entità: Studente, Corso
* Relazione: frequenta (N:M)

#### **Implementazione:**

Si crea una **tabella intermedia** che collega le due entità.

sql

CopiaModifica

CREATE TABLE Studente (

ID INT PRIMARY KEY,

Nome VARCHAR(50)

);

CREATE TABLE Corso (

ID INT PRIMARY KEY,

NomeCorso VARCHAR(50)

);

CREATE TABLE Frequenza (

IDStudente INT,

IDCorso INT,

PRIMARY KEY (IDStudente, IDCorso),

FOREIGN KEY (IDStudente) REFERENCES Studente(ID),

FOREIGN KEY (IDCorso) REFERENCES Corso(ID)

);

Questa tabella può anche contenere **attributi della relazione**, come la data di iscrizione (Dello studente al corso) o il voto finale (Dello studente al corso).

## **11. Consigli per una buona progettazione concettuale**

* Usa **nomi significativi** per entità e attributi.
* Elimina le **ridondanze** (ripetizioni inutili).
* Verifica che ogni entità abbia una **chiave primaria**.
* Identifica correttamente i **tipi di relazione**.
* Usa **diagrammi E-R** per visualizzare la struttura.
* Pensa a **casi d'uso reali** (es. gestione scuola, biblioteca, e-commerce).

## 

## **12 UN ESEMPIO: Relazioni tra Docenti e Studenti**

### **Contesto scolastico: Istituto Tecnico Industriale**

### 🔹 Relazione 1-a-N tra Docente di Informatica e Studente

### **Situazione:**

Il docente di informatica è **tutor del progetto di esame** di ogni studente della classe quinta.  
 Ogni **studente** ha **un solo tutor** (che è il docente di informatica),  
 ma **lo stesso docente** segue **tutti gli studenti** della sua classe.

### **➤ Tipo di relazione:**

**DocenteInformatica — (1:N) — Studente**

* Un **docente** → segue **più studenti**
* Ogni **studente** → ha **un solo tutor**

### **➤ Modello concettuale:**

* Entità: DocenteInformatica, Studente
* Relazione: tutoring

### **➤ Implementazione:**

sql

CopiaModifica

CREATE TABLE DocenteInformatica (

ID INT PRIMARY KEY,

Nome VARCHAR(50)

);

CREATE TABLE Studente (

ID INT PRIMARY KEY,

Nome VARCHAR(50),

IDDocente INT,

FOREIGN KEY (IDDocente) REFERENCES DocenteInformatica(ID)

);

### 🔸 Relazione N-a-M tra Docente Generico e Studente

### **Situazione:**

Ogni **studente** partecipa a **più corsi** (es. Italiano, Matematica, Fisica, Informatica),  
 e ogni **docente** insegna a **più studenti** in più classi.

### **➤ Tipo di relazione:**

**Docente — (N:M) — Studente**

* Un **docente** → ha **più studenti**
* Uno **studente** → ha **più docenti**

### **➤ Modello concettuale:**

* Entità: Docente, Studente
* Relazione: insegna\_a

#### **Nella relazione possiamo aggiungere attributi come:**

* Materia
* Classe
* AnnoScolastico

### **➤ Implementazione (con tabella intermedia):**

sql

CopiaModifica

CREATE TABLE Docente (

ID INT PRIMARY KEY,

Nome VARCHAR(50)

);

CREATE TABLE Studente (

ID INT PRIMARY KEY,

Nome VARCHAR(50)

);

CREATE TABLE Insegnamento (

IDDocente INT,

IDStudente INT,

Materia VARCHAR(50),

AnnoScolastico VARCHAR(9),

PRIMARY KEY (IDDocente, IDStudente, Materia),

FOREIGN KEY (IDDocente) REFERENCES Docente(ID),

FOREIGN KEY (IDStudente) REFERENCES Studente(ID)

);

### 🔁 Riepilogo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Relazione** | **Tipo** | **Chiave esterna diretta?** | **Tabella intermedia?** | **Esempio reale** |
| Docente di Informatica ↔ Studente | 1:N | Sì (nella tabella Studente) | No | Tutoraggio di progetti |
| Docente ↔ Studente | N:M | No | Sì (Insegnamento) | Materie scolastiche in un intero anno |

## 13 UN ESEMPIO DI SIMULAZIONE DELLA PROVA DELLA MATURITA’

Ecco un esempio strutturato per la **seconda prova di Informatica** della Maturità ITIS 2026, basato su una piattaforma per il riconoscimento delle fake news.

I docenti possono caricare le notizie. Ognuno ne può caricare quante ne vuole. Toccherà poi agli studenti verificarle e studenti diversi possono lavorare sulla stessa notizia per decidere come classificarla.

### 🎯 Traccia – Sintesi

Gli studenti dovevano progettare una piattaforma web in cui vengono raccolte e etichettate notizie (fake news o veritiere). Compito includeva:

1. Schema concettuale e logico
2. Definizione delle tabelle SQL con vincoli
3. Interrogazioni SQL per analisi statistiche su etichettature
4. (Opzionale) elementi di front-end/back-endPHP‑MySQL

Queste indicazioni sono state confermate dalle fonti: la prova riguardava dataset di notizie, etichette, e query SQL su conteggi e filtraggi [toghiri.github.io](https://toghiri.github.io/media/pdf/scuola/simulazioni_esame/2023/soluzione_2_2023.pdf?utm_source=chatgpt.com).

### 📐 Modello concettuale (ER)

**Entità principali:**

* **Notizia** *(ID, Titolo, Testo, DataPubblicazione, Argomento…)*
* **Studente** *(ID, Nome, Classe…)*
* **Docente** *(ID, Nome, Materia…)*

**Relazioni:**

* **Etichetta** *(Studente–Notizia): N:M con attributo “label” (fake/true) e DataEtichetta*
* **Assegna** *(Docente–Notizia): 1:N se ogni notizia assegnata da un solo docente*

### Schema logico e tabelle SQL

sql

CopiaModifica

CREATE TABLE Notizia (

ID INT PRIMARY KEY AUTO\_INCREMENT,

Titolo VARCHAR(200),

Testo TEXT,

DataPubblicazione DATE,

Argomento VARCHAR(100)

);

CREATE TABLE Studente (

ID INT PRIMARY KEY,

Nome VARCHAR(100),

Classe VARCHAR(5)

);

CREATE TABLE Docente (

ID INT PRIMARY KEY,

Nome VARCHAR(100),

Materia VARCHAR(50)

);

CREATE TABLE Etichetta (

IDStudente INT,

IDNotizia INT,

Label ENUM('fake','true'),

DataEtichetta DATETIME,

PRIMARY KEY (IDStudente, IDNotizia),

FOREIGN KEY (IDStudente) REFERENCES Studente(ID),

FOREIGN KEY (IDNotizia) REFERENCES Notizia(ID)

);

-- relazione 1:N docente→notizia

ALTER TABLE Notizia ADD COLUMN IDDocente INT,

ADD FOREIGN KEY (IDDocente) REFERENCES Docente(ID);

### Query SQL richieste

Ecco alcune interrogazioni tipiche:

**a) Numero di notizie false etichettate per studente:**

sql

CopiaModifica

SELECT s.Nome, COUNT(\*) AS Falsi

FROM Etichetta e

JOIN Studente s ON e.IDStudente = s.ID

WHERE e.Label = 'fake'

GROUP BY s.ID;

**b) Notizie di un certo argomento ordinate per numero di etichette false:**

sql

CopiaModifica

SELECT n.Titolo,

SUM(e.Label='fake') AS NumFake

FROM Notizia n

LEFT JOIN Etichetta e ON e.IDNotizia = n.ID

WHERE n.Argomento = 'Politica'

GROUP BY n.ID

ORDER BY NumFake DESC;

**c) Numero di studenti che hanno etichettato ciascuna notizia:**

sql

CopiaModifica

SELECT n.Titolo, COUNT(e.IDStudente) AS NumStudenti

FROM Notizia n

LEFT JOIN Etichetta e ON e.IDNotizia = n.ID

GROUP BY n.ID;

### Spiegazione del database progettato

* **Relazione N:M** tra studenti e notizie → la tabella Etichetta raccoglie ogni interazione e memorizza il tipo e data dell'etichettatura.
* **Relazione 1:N** docente−notizia → ogni notizia è caricata da un solo docente, ma ciascun docente può caricarne molte.
* Il database rispetta la normalizzazione, evitando ridondanze, e garantisce integrità referenziale.

Riassunto del tuo scenario:

1. **Ogni docente può caricare più notizie** 👉 relazione **1:N** tra Docente e Notizia
2. **Ogni notizia può essere analizzata da più studenti**
3. **Ogni studente può lavorare su più notizie** 👉 relazione **N:M** tra Studente e Notizia, gestita dalla tabella Etichetta

Come viene rappresentato nello schema SQL

🔹 Docente — 1:N — Notizia

Ogni notizia ha un solo docente autore.  
 Si realizza con una **chiave esterna IDDocente** nella tabella Notizia:

sql

CopiaModifica

CREATE TABLE Docente (

ID INT PRIMARY KEY,

Nome VARCHAR(100)

);

CREATE TABLE Notizia (

ID INT PRIMARY KEY AUTO\_INCREMENT,

Titolo VARCHAR(200),

Testo TEXT,

DataPubblicazione DATE,

Argomento VARCHAR(100),

IDDocente INT,

FOREIGN KEY (IDDocente) REFERENCES Docente(ID)

);

🔸 Studente — N:M — Notizia (tramite Etichetta)

Più studenti possono esaminare la stessa notizia  
 e ogni studente può verificare più notizie.  
 Si usa una tabella di **associazione** (Etichetta) con attributi:

sql

CopiaModifica

CREATE TABLE Studente (

ID INT PRIMARY KEY,

Nome VARCHAR(100),

Classe VARCHAR(10)

);

CREATE TABLE Etichetta (

IDStudente INT,

IDNotizia INT,

Label ENUM('fake', 'true') NOT NULL,

DataEtichetta DATETIME,

PRIMARY KEY (IDStudente, IDNotizia),

FOREIGN KEY (IDStudente) REFERENCES Studente(ID),

FOREIGN KEY (IDNotizia) REFERENCES Notizia(ID)

);

🧠 Spiegazione logica

* La tabella Etichetta rappresenta **l’azione dello studente** su una notizia → è una **relazione N:M con attributi**.
* La tabella Notizia ha un campo IDDocente che lega ciascuna notizia al docente che l’ha caricata.
* Puoi anche estendere Etichetta con un campo come Commento o Punteggio, se desideri arricchire l'interazione.

## 14 NORMALIZZAZIONE DI UN DATABASE

La **normalizzazione** è un processo che serve a **organizzare i dati in un database relazionale** per:

* Eliminare **ridondanze** (duplicazioni di dati)
* Evitare **anomalie** (di aggiornamento, inserimento, cancellazione)
* Rendere il database **più efficiente e coerente**

Il processo si sviluppa in **forme normali**: ogni forma normale risolve un determinato tipo di problema.

### 🔹 PRIMA FORMA NORMALE (1NF)

**📌 Concetto: dati atomici**

I valori in ogni **cella della tabella** devono essere **atomici**, cioè **non suddivisibili ulteriormente**.

**❌ Esempio NON in 1NF:**

|  |  |
| --- | --- |
| **Studente** | **Materie** |
| Mario | Informatica, Matematica |

→ la colonna **"Materie"** contiene una **lista di valori** → ❌ non è atomica

**✅ Forma corretta (1NF):**

|  |  |
| --- | --- |
| **Studente** | **Materia** |
| Mario | Informatica |
| Mario | Matematica |

→ ogni riga contiene **un solo valore per attributo**

### 🔸 SECONDA FORMA NORMALE (2NF)

**📌 Concetto: tutti gli attributi devono dipendere dall’intera chiave primaria**

Si applica **solo se la tabella ha una chiave composta** (cioè più colonne che identificano univocamente una riga).  
 La **2NF** elimina le **dipendenze parziali**, separando le informazioni che **dipendono solo da una parte della chiave**.

**🔁 Collegamento con relazione 1:N:**

La normalizzazione spesso porta a **separare entità** collegate da una relazione 1:N.

**❌ Esempio NON in 2NF:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **IDStudente** | **IDCorso** | **NomeStudente** | **NomeCorso** | **Voto** |
| 1 | 101 | Mario | Informatica | 8 |

→ NomeStudente dipende solo da IDStudente,  
 → NomeCorso dipende solo da IDCorso,  
 → ma la chiave primaria è (IDStudente, IDCorso)

**✅ Forma corretta (2NF):**

**Tabella Studente:**

|  |  |
| --- | --- |
| **IDStudente** | **NomeStudente** |
| 1 | Mario |

**Tabella Corso:**

|  |  |
| --- | --- |
| **IDCorso** | **NomeCorso** |
| 101 | Informatica |

**Tabella Iscrizione:**

|  |  |  |
| --- | --- | --- |
| **IDStudente** | **IDCorso** | **Voto** |
| 1 | 101 | 8 |

→ abbiamo ottenuto **una struttura relazionale 1:N** tra Studente ↔ Iscrizione ↔ Corso

### 🔺 TERZA FORMA NORMALE (3NF)

**📌 Concetto: nessuna dipendenza transitiva**

Un attributo **non chiave** non deve **dipendere da un altro attributo non chiave**, ma **solo dalla chiave primaria**.

**🔁 Collegamento con relazione N:M:**

La normalizzazione in 3NF porta spesso alla **creazione di una tabella intermedia** per rappresentare correttamente una relazione molti-a-molti.

**❌ Esempio NON in 3NF:**

|  |  |  |
| --- | --- | --- |
| **IDStudente** | **IDCorso** | **NomeDocente** |
| 1 | 101 | Rossi |

→ NomeDocente dipende da IDCorso, **non direttamente** dalla chiave composta → dipendenza **transitiva**

**✅ Forma corretta (3NF):**

**Tabella Corso:**

|  |  |  |
| --- | --- | --- |
| **IDCorso** | **NomeCorso** | **IDDocente** |
| 101 | Informatica | 10 |

**Tabella Docente:**

|  |  |
| --- | --- |
| **IDDocente** | **NomeDocente** |
| 10 | Rossi |

**Tabella Iscrizione:**

|  |  |  |
| --- | --- | --- |
| **IDStudente** | **IDCorso** | **Voto** |
| 1 | 101 | 8 |

→ così abbiamo normalizzato e ottenuto una struttura chiara con relazione **N:M** tra Studente e Corso, e **1:N** tra Docente e Corso.

### 🧠 Riepilogo visivo

|  |  |  |
| --- | --- | --- |
| **Forma Normale** | **Cosa garantisce** | **Collegamento** |
| **1NF** | Dati atomici | Nessuna lista o array nei campi |
| **2NF** | Nessuna dipendenza parziale | Nasce spesso una **1:N** (es. Studente → Iscrizione) |
| **3NF** | Nessuna dipendenza transitiva | Si separano entità collegate in **N:M** (es. Studente ↔ Corso) |

## 15 SINTASSI SQL – GUIDA PRATICA CON ESEMPI

Il linguaggio SQL è diviso in tre grandi categorie operative:

1. 🔧 **DDL** – *Data Definition Language*
2. 📝 **DML** – *Data Manipulation Language*
3. 🔍 **DQL** – *Data Query Language*

Utilizzeremo un esempio concreto basato su una **piattaforma scolastica** in cui si gestiscono **studenti, corsi e voti**.

### ✅ 1. DDL – Data Definition Language

Il DDL serve per **definire la struttura del database**: creare, modificare o cancellare tabelle e relazioni.

✨ Comandi principali:

* CREATE TABLE
* ALTER TABLE
* DROP TABLE

#### 📌 Esempio: creare le tabelle Studente e Corso

sql

CopiaModifica

CREATE TABLE Studente (

ID INT PRIMARY KEY AUTO\_INCREMENT,

Nome VARCHAR(50),

Cognome VARCHAR(50),

Classe VARCHAR(10)

);

CREATE TABLE Corso (

ID INT PRIMARY KEY AUTO\_INCREMENT,

NomeCorso VARCHAR(100),

Docente VARCHAR(100)

);

#### ➕ Aggiunta colonna alla tabella

sql

CopiaModifica

ALTER TABLE Studente ADD COLUMN Email VARCHAR(100);

#### ❌ Eliminazione tabella

sql

CopiaModifica

DROP TABLE Corso;

### ✍️ 2. DML – Data Manipulation Language

Il DML viene usato per **inserire, modificare e cancellare i dati** dentro le tabelle.

✨ Comandi principali:

* INSERT INTO
* UPDATE
* DELETE

#### 📌 Inserimento di dati

sql

CopiaModifica

INSERT INTO Studente (Nome, Cognome, Classe)

VALUES ('Mario', 'Rossi', '5AINF');

#### 🛠️ Modifica di un dato

sql

CopiaModifica

UPDATE Studente

SET Classe = '5BINF'

WHERE ID = 1;

#### 🗑️ Cancellazione di un dato

sql

CopiaModifica

DELETE FROM Studente

WHERE ID = 1;

⚠️ Attenzione: se il WHERE è assente, **si cancellano tutte le righe!**

### 🔎 3. DQL – Data Query Language

Il DQL serve per **interrogare** i dati presenti nel database, cioè per **estrarre informazioni**.

✨ Comando principale:

* SELECT

#### 📌 Esempi:

**a) Visualizzare tutti gli studenti**

SELECT \* FROM Studente;

**b) Visualizzare solo i nomi e la classe**

SELECT Nome, Classe FROM Studente;

**c) Studenti della classe 5AINF**

SELECT \* FROM Studente

WHERE Classe = '5AINF';

**d) Ordinare gli studenti per cognome**

SELECT \* FROM Studente

ORDER BY Cognome ASC;

**e) Contare quanti studenti ci sono in una classe**

SELECT COUNT(\*) AS NumeroStudenti

FROM Studente

WHERE Classe = '5AINF';

### 🧠 Riepilogo

|  |  |  |  |
| --- | --- | --- | --- |
| **Categoria** | **Significato** | **Cosa fa** | **Esempi** |
| **DDL** | Data Definition Language | Definisce la struttura | CREATE, ALTER, DROP |
| **DML** | Data Manipulation Language | Inserisce/modifica dati | INSERT, UPDATE, DELETE |
| **DQL** | Data Query Language | Interroga dati | SELECT |

## 16 Cos’è XAMPP

**XAMPP** è un **pacchetto software gratuito** che installa su un computer tutto il necessario per **sviluppare applicazioni web in locale**, cioè **senza bisogno di una connessione a Internet** o di un server esterno.

### **✅ Cosa include XAMPP?**

|  |  |
| --- | --- |
| **Componente** | **Descrizione** |
| **X** | Sta per "Cross-platform" (funziona su Windows, Linux, Mac) |
| **A** | **Apache** – Web server che gestisce le richieste HTTP |
| **M** | **MySQL** o MariaDB – Database relazionale |
| **P** | **PHP** – Linguaggio per creare pagine web dinamiche |
| **P** | **Perl** – Altro linguaggio per il web, meno usato oggi |

📌 **In sintesi**: XAMPP ti permette di creare e testare un sito web con database **sul tuo computer**, senza dover pubblicare nulla online.

### 🗃️ Cos’è MySQL

#### 🧠 1. MySQL come DBMS (Database Management System)

Un **DBMS** è un **programma** che serve a **creare, gestire, modificare** e **interrogare** un **database**.  
 **MySQL** è un **DBMS relazionale**, cioè i dati sono organizzati in **tabelle collegate tra loro da relazioni**.

**✅ Cosa fa MySQL come DBMS?**

* Crea tabelle, relazioni, indici, viste
* Gestisce utenti e permessi
* Gestisce le **operazioni CRUD** (Create, Read, Update, Delete)
* Risponde alle **query SQL** inviate da applicazioni PHP, Python, Java ecc.

💡 Un DBMS è come un **"motore"** che esegue i comandi SQL e gestisce i dati.

MySQL come Database

Nel linguaggio quotidiano, spesso si dice “database MySQL” per indicare **l’insieme dei dati** memorizzati in MySQL.

Ma tecnicamente:

* **Il DBMS è il software** MySQL
* **Un database** è un **insieme organizzato di tabelle e dati** (es. scuola, negozio, biblioteca...)

### **🔍 Esempio:**

Immaginiamo che tu stia lavorando su un gestionale scolastico.

* **MySQL (DBMS)** è il motore che esegue i comandi SQL
* **Il database Scuola** contiene tabelle come:

plaintext

CopiaModifica

Scuola

├── Studenti

├── Docenti

├── Corsi

├── Iscrizioni

Quindi, nel tuo progetto:

* Scrivi codice PHP o SQL → viene inviato al **DBMS MySQL**
* Il DBMS lo esegue → sui dati nel **database specifico**

### Come funziona nella pratica con XAMPP

1. Avvii **XAMPP Control Panel**
2. Avvii **Apache** (per il server web)
3. Avvii **MySQL** (per il database)
4. Vai su http://localhost/phpmyadmin → interfaccia grafica per gestire i database
5. Crei un database (es. scuola)
6. All’interno crei tabelle (studenti, docenti, etc.)
7. Lanci query SQL oppure le gestisci da una web app in PHP

### 🧠 Riepilogo

|  |  |
| --- | --- |
| **Termine** | **Significato** |
| **XAMPP** | Pacchetto completo per creare applicazioni web in locale |
| **MySQL DBMS** | Software che gestisce i dati: crea, modifica, interroga i database |
| **MySQL Database** | L’insieme concreto di tabelle e dati gestiti dal DBMS |

## 18 Apache, il Web Server, e l’architettura client-server

### 🔥 Cos’è Apache?

**Apache** è un **web server**, ovvero un programma che:

* Riceve **richieste HTTP** (es. da un browser)
* Risponde inviando **pagine web** (HTML, PHP, CSS, JS…)
* Viene **installato su un server** connesso alla rete

📌 **Non crea pagine web**: le **ospita** e le **serve** quando richieste.

### 🖥️ Dove si trova Apache?

**✔️ In produzione (cioè online)**

Apache è **installato su un server pubblico** (es. Aruba, OVH, AWS) dove pubblichiamo il nostro sito.

* Apache contiene:  
  + Pagine **HTML**
  + File **PHP**
  + Script **JS**, **CSS**, immagini…
* Un **altro server** (o lo stesso) ospita **MySQL** come **DBMS**
* L’utente finale (**client**) usa il **browser** per navigare

🔁 Il browser del client invia richieste → Apache risponde → PHP si collega al database

**🧪 In sviluppo (cioè in locale)**

Per **programmare** il sito web, possiamo simulare tutto **sul nostro computer** usando **XAMPP**:

* Apache è **locale** (in esecuzione nel nostro PC)
* MySQL è **locale**
* Le pagine sono salvate nella cartella htdocs di XAMPP
* Apriamo il browser e andiamo su:  
   👉 http://localhost/miosito

Questo è utile solo per **test e sviluppo**, **non è visibile da altri su Internet**

### 🧭 Architettura Client - Server in sintesi

|  |  |  |
| --- | --- | --- |
| **Ruolo** | **Cosa fa** | **Dove si trova** |
| **Client (es. Chrome)** | Invia richieste HTTP (es. clic su un link) | PC dell’utente |
| **Apache (Web Server)** | Riceve la richiesta, invia HTML o esegue PHP | Server remoto o locale |
| **PHP** | Elabora i dati e genera la pagina da mostrare | Sul server, dentro Apache |
| **MySQL (DBMS)** | Memorizza ed elabora i dati | Server dedicato o stesso di Apache |

### 🔁 Esempio reale (produzione)

1. Un utente visita www.miosito.it
2. Il browser invia una richiesta al **server Apache**
3. Apache esegue un file PHP che interroga il **database MySQL** (su un altro server)
4. PHP riceve i dati, crea la pagina HTML, e Apache la invia al browser

### 🧪 Esempio in locale con XAMPP

1. Lo sviluppatore avvia **Apache e MySQL da XAMPP**
2. Scrive il codice PHP + HTML + SQL
3. Lo salva in htdocs/
4. Apre il browser e digita localhost/miosito
5. Apache locale esegue i file e interroga MySQL locale

### 📌 Conclusione

* **Apache** è un **web server** da installare su un server (locale o online)
* Ospita le pagine PHP e HTML e **risponde ai browser**
* In **produzione**, il server Apache può essere **diverso dal server MySQL**
* In **locale**, XAMPP simula tutto per **sviluppare e testare**