Linguaggio SQL (parte 1)

Mattia Cozzi
cozzimattia@gmail.com

a.s. 2024/2025

Contenuti

Basi di dati

Nelle applicazioni informatiche sono presenti informazioni che è necessario memorizzare in modo permanente per renderle utilizzabili per elaborazioni. Inoltre, quando molti utenti lavorano su un insieme di dati, è necessario avere una sola copia dei dati, sempre aggiornata, che consenta l'accesso simultaneo a più utenti. Questo compito è svolto dai database (basi di dati).

Le basi di dati sono raccolte di dati progettati in modo tale da poter essere utilizzati in maniera ottimizzata da diverse applicazioni e diversi utenti.

Modello Entità-Relazione

Il modello E-R permette di modellare graficamente il mondo reale sotto forma di entità e relazioni tra esse.

Le entità sono gli oggetti principali su cui vengono raccolte le informazioni. Ogni entità rappresenta graficamente un oggetto, concreto o astratto, del mondo reale.

Ogni entità avrà un nome, che permette di identificare ogni istanza (ogni "esemplare") di quella classe.

Le relazioni tra entità verranno rappresentate mediante linee che collegano le entità.

Istanze

Ogni istanza di una certa entità è caratterizzato da un insieme di valori che descrivono le sue proprietà.

Tutte le istanze di una certa entità hanno gli stessi attributi, ma con valori diversi per poterle distinguere.

Ad esempio, per l'entità "Studente", gli attributi possono essere "Nome", "Cognome", "Codice fiscale", "Data di nascita", "Sezione", ecc.

Attributi

Per l'entità "Persona":

Nome	stringa(20), obbligatorio, non NULL
Cognome	stringa(20), obbligatorio, non NULL
CodFiscale	stringa(16)
TitoloStudio	stringa(50)
DataNascita	giorno - mese - anno
Peso	numerico
AnniServizio	numerico

Attributi chiave

La chiave primaria è un identificatore univoco (ID) di un'istanza di un'entità.

Esempio: la numerazione sequenziale delle tessere rilasciate da una associazione.

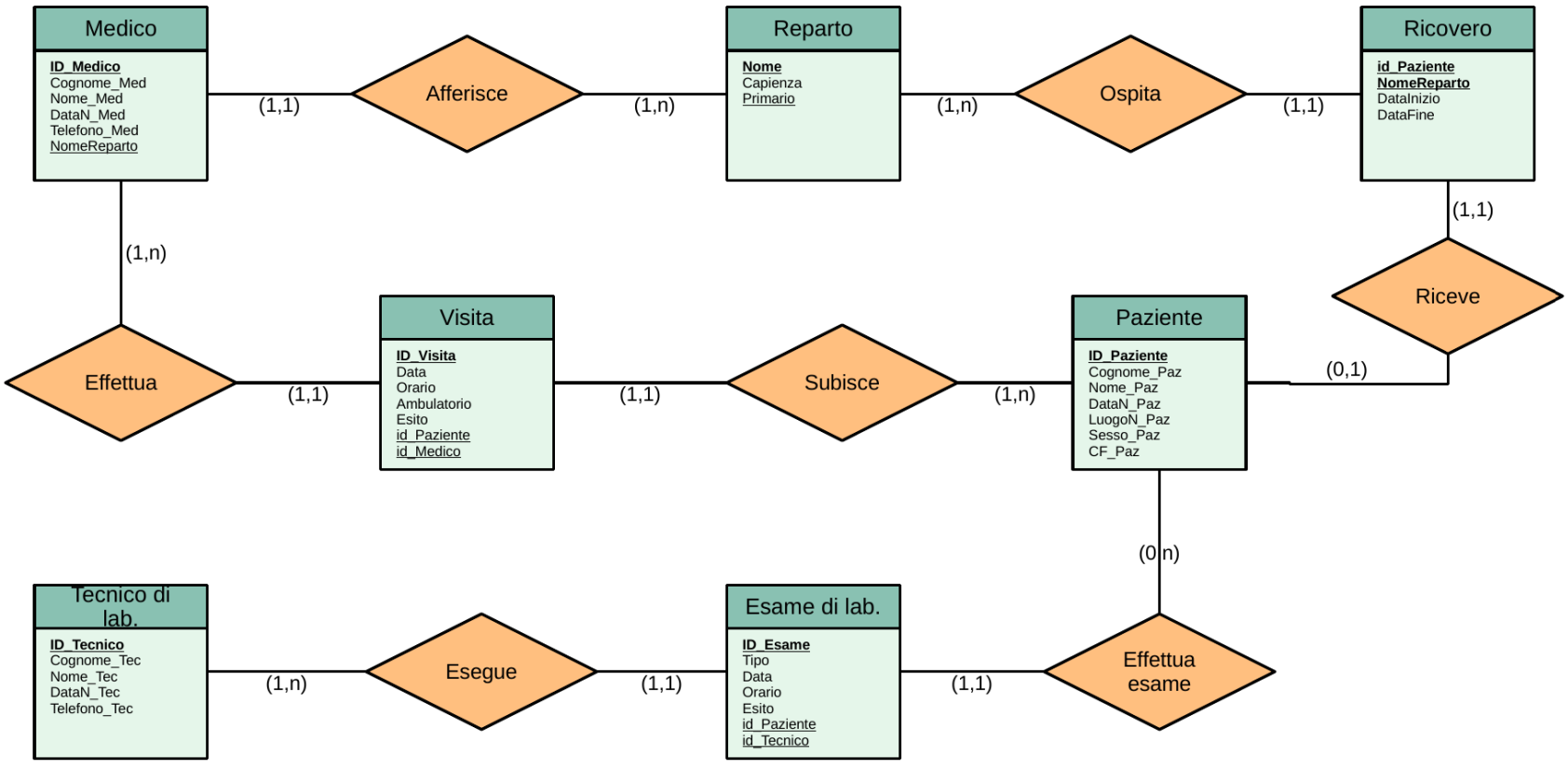
Alle chiavi primarie viene solitamente dato un nome che inizia con ID_ (ID_Studente, ID_Auto).

Chiavi esterne

Le chiavi esterne (*foreign keys*) sono utilizzate per realizzare i collegamenti tra entità, risolvendo quindi i collegamenti uno-a-molti.

Se le chiavi esterne sono chiavi artificiali (codici), vengono nominate in modo simile a quanto si fa per le chiavi primarie, con il prefisso id_ (minuscolo).

Esempio di diagramma E-R (schema concettuale)



Realizzazione dello schema logico

Il modello (o schema) logico viene utilizzato come input per la progettazione vera e propria del database: deve avere quindi il massimo livello di dettaglio e precisione possibile.

Deve contenere tutte le informazioni necessarie per definire le tabelle, riportando la descrizione puntuale e completa del significato di ogni dato memorizzato.

Lo schema logico trasforma le informazioni del modello concettuale in un formato efficiente.

Schema logico del database ospedaliero

Medico	ID_Medico (pk), Cognome_Med, Nome_Med, DataN_Med, Telefono_Med, NomeReparto (fk);
Paziente	ID_Paziente (pk), Cognome_Paz, Nome_Paz, DataN_Paz, LuogoN_Paz, Sesso_Paz, CF_Paz;
Reparto	Nome (pk), Capienza, Primario (fk);
Ricovero	id_Paziente (pk), NomeReparto (pk), DataInizio, DataFine;
Visita	ID_Visita (pk), Data, Orario, Ambulatorio, Esito, id_Paziente (fk), id_Medico (fk);
EsameLab	ID_Esame (pk), Tipo, Data, Orario, Esito, id_Paziente (fk), id_Tecnico (fk);
TecnicoLab	ID_Tecnico (pk), Cognome_Tec, Nome_Tec, DataN_Tec, Telefono_Tec;

Tabelle relazionali

Il modello relazionale, introdotto da E.F. Codd nel 1970, presenta i dati nella forma di tabelle. Le colonne delle tabelle rappresentano campi o proprietà, le righe rappresentano i diversi record.

Ogni tabella ha:

- un nome per ogni colonna;
- un elenco di colonne di cui vengono specificati i tipi;
- una chiave primaria che identifica univocamente ogni riga della tabella.

Due tabelle sono collegate tramite chiavi esterne.

Le colonne possono anche essere chiamate “campi” o “attributi”, mentre le righe possono anche essere chiamate “record” o “tuple”.

Traduzione (1)

Seguiamo queste cinque regole:

1. ogni entità diventa una tabella;
2. ogni attributo dell'entità diventa una colonna della tabella;
3. ogni istanza di un'entità diventa una riga della tabella;
4. la chiave primaria dell'entità è l'identificatore univoco delle righe di una tabella;
5. le relazioni sono rappresentate con chiavi esterne che collegano righe di due diverse tabelle.

Traduzione (2)

Si è soliti inserire la chiave primaria nella prima colonna a sinistra, mentre le chiavi esterne nelle ultime colonne a destra.

Tabella Clienti

ID_Cliente	Ragione_Sociale	Indirizzo	Partita_IVA
120	Rossi SRL	Via Roma, 3	10203040501
220	Verdi snc	Via Torino, 23	20304040501
333	Gialli SPA	Via Napoli, 11	40503040501
520	Rossi SRL	Via Genova, 13	30503040501

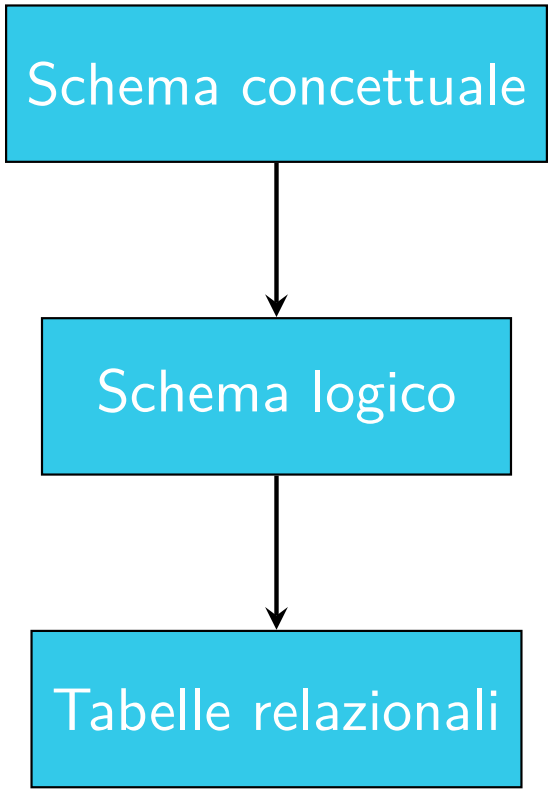
Istanze della tabella – Tuple – Record

Attributi – Campi

Tabella Fatture

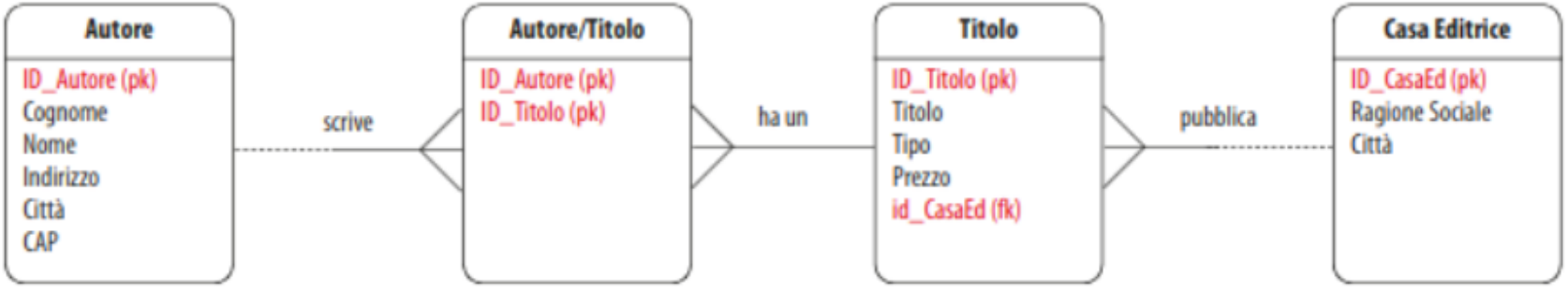
Numero	Data	Importo	id_Cliente
70	01/04/2018	1200,20	120
71	03/04/2018	1450,29	520
72	04/04/2018	1200,20	120
73	05/04/2018	2300,45	220
74	06/04/2018	1200,20	120

Fasi della realizzazione di un database



Dallo schema concettuale al modello relazionale (1)

Per un database bibliografico con il seguente schema concettuale:



si ottiene il seguente schema logico:

Autori (ID_Autore(pk), Cognome, Nome, Indirizzo, Città, CAP)
Titoli (ID_Titolo(pk), Titolo, Tipo, Prezzo, id_CasaEd(fk))
Case Editrici (ID_CasaEd(pk), Ragione Sociale, Città)
Autori_Titoli (ID_Autore(pk), ID_Titolo(pk))

Dallo schema concettuale al modello relazionale (2)

Lo schema logico precedente diventa poi un modello relazionale:

AUTORI

ID_Autore	Cognome	Nome	Indirizzo	Città	CAP
172	Bianchi	Mario	Via Garibaldi, 10	Brescia	25100
156	Cerreti	Giovanni	Via Carducci, 12	Milano	20100
268	Rossigni	Giacomo	Piazza Caduti, 20	Torino	10100
884	Tacchi	Alfio	Borgo Trento	Treviso	31100
343	Bertozzi	Filippo	Via del lago, 9	Arenzano	16011
298	Sommo	Norberto	Largo Roma, 3	Como	22100

AUTORI_TITOLI

ID_Autore	ID_Titolo
172	7771010
156	7771213
268	7772123
884	7773244
343	7771356
298	7774744

TITOLI

ID_Titolo	Titolo	Tipo	Prezzo	id_CasaEd
7771010	*C++*	Informatica	20.00	91234
7771213	*La cucina mediterranea*	Cucina	18.50	92233
7772123	*Aumentare gli utili*	Economia	36.70	94354
7773244	*L'ultima neve*	Thriller	15.20	95555
7771356	*Stress: il nostro nemico*	Psicologia	18.40	94354
7774744	*Le leggi di Mendel*	Biologia	21.00	95555

CASE EDITRICI

ID_CasaEd	Ragione Sociale	Città
91234	Tecniche OK	Brescia
92233	Nuove Gastronomie	Milano
94354	Hoepli	Milano
94354	Mondo Libri	Torino
95555	New Gothics	Treviso

Architettura client-server

Nel nostro corso utilizzeremo un database relazionale con architettura client-server.

Per funzionare richiede quindi un server in funzione (in rete o in locale, cioè sul nostro computer) e un client per poter interagire col server.

Esistono client a riga di comando, client visuali oppure client web (che girano all'interno di una pagina web).

Il nostro server

Come server utilizzeremo il software open source XAMPP, che gira su tutte le piattaforme.

https://www.apachefriends.org/it/index.html

Alcuni moduli di XAMPP:

- Apache, un webserver locale, permette di non caricare i nostri progetti su un server remoto ma farli girare in locale;
- MySQL, un sistema di gestione di database relazionali in linguaggio PHP; richiede Apache;
- FileZilla, client FTP, per trasferire file sui server.

Avvio di XAMPP

Una volta installato, possiamo avviare XAMPP dal menu delle applicazioni (Windows e MacOS) oppure usare il comando Linux:

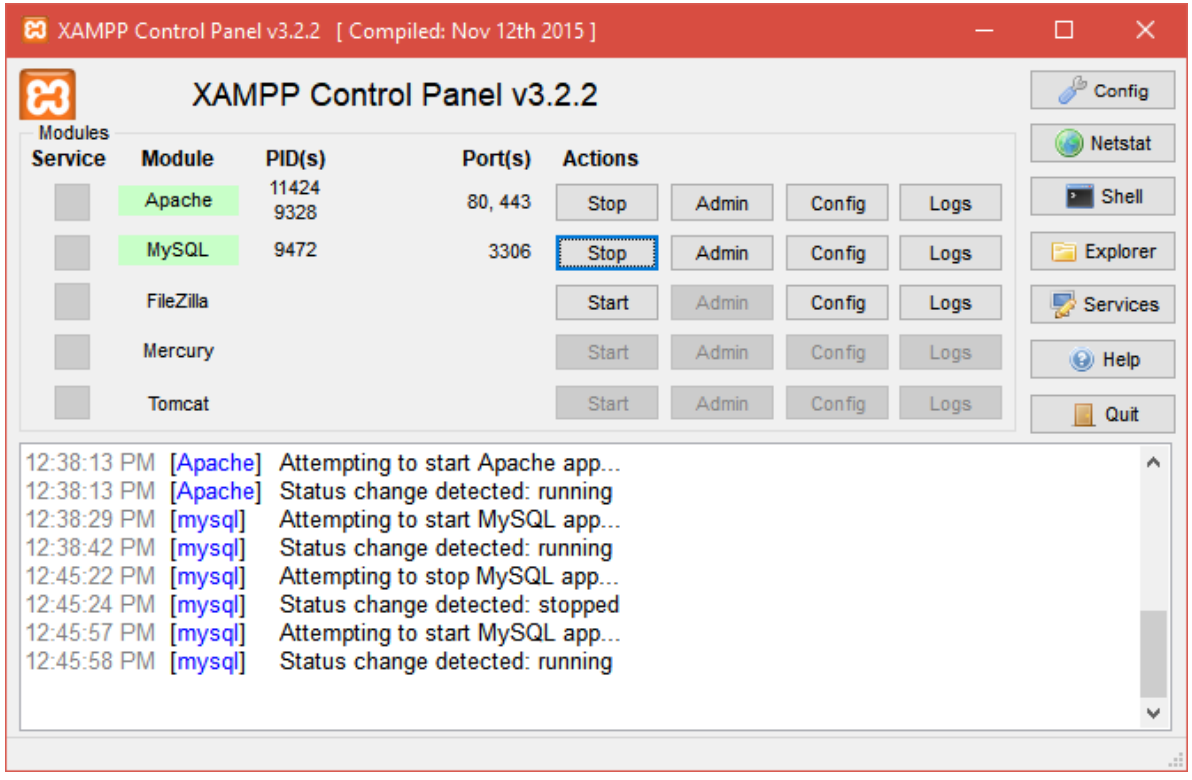
```
sudo /opt/lampp/manager-linux-x64.run
```

Sempre su Linux, per gestire XAMPP da linea di comando, diamo:

```
sudo /opt/lampp/xampp
```

Avvio del server

Per avviare il server web e MySQL clicchiamo su “Start” per entrambi i servizi:



CRUD

L'acronimo CRUD sta per:

- Create, inserimento di nuovi dati;
- Read, accesso ai dati;
- Update, modifica dei dati;
- Delete, eliminazione dei dati.

Questo acronimo indica le quattro operazioni fondamentali che possiamo compiere su un database.

In questo corso impareremo a gestire un database tramite Python.

Cosa è SQL

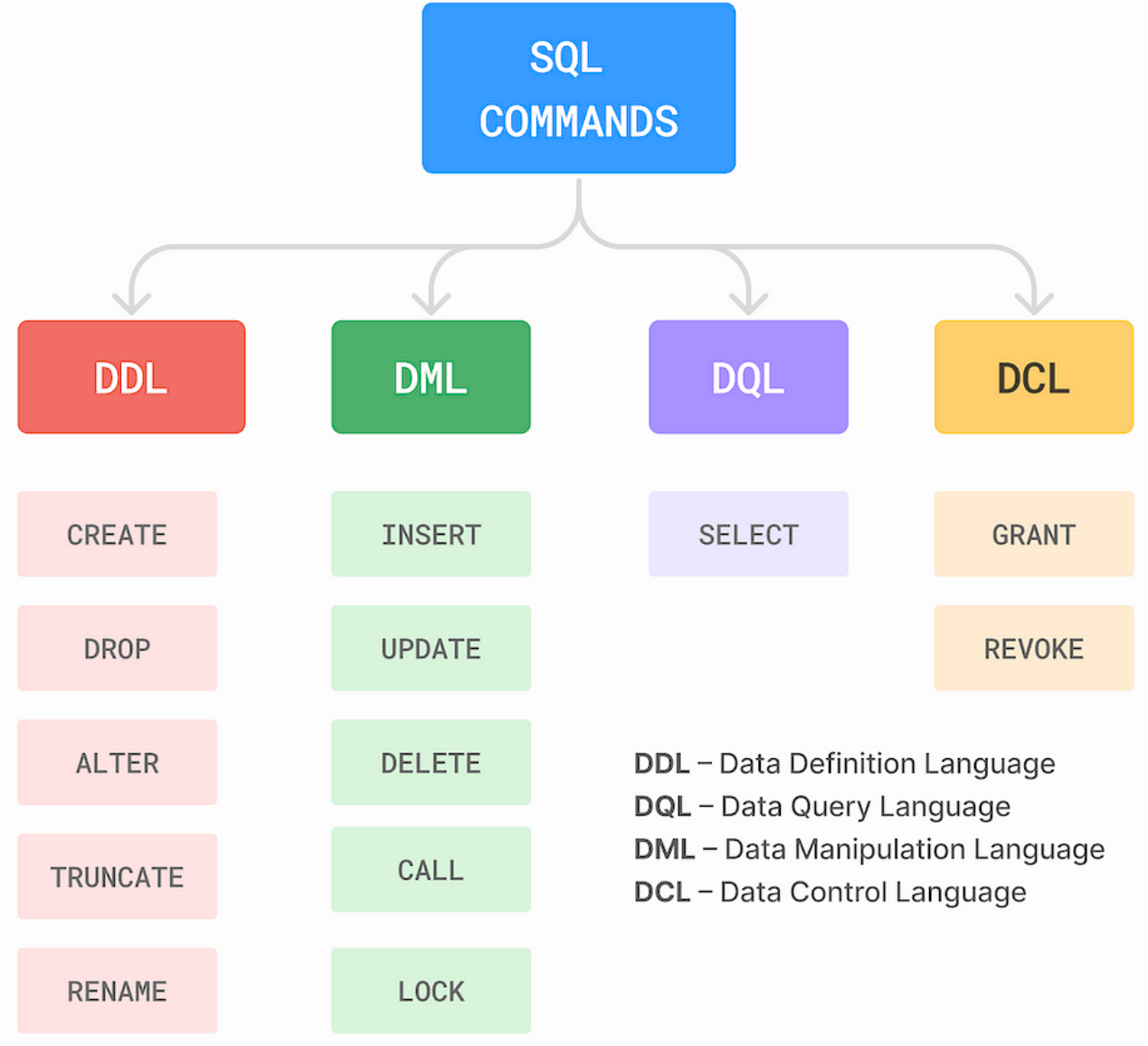
SQL (*Structured Query Language*) è un linguaggio per la manipolazione dei dati di un database.

Può essere suddiviso in quattro sottolinguaggi:

- DDL (*Data Description Language*);
- DQL (*Data Query Language*);
- DML (*Data Manipulation Language*);
- DCL (*Data Control Language*).

Noi ci occuperemo dei primi tre.

Schema comandi di SQL



Struttura di un comando SQL

Un comando in SQL si suddivide in:

- costrutto, cioè il tipo di operazione effettuata, come `SELECT`, `INSERT` o `CREATE`;
- parametri, come `FROM`, `WHERE` o `IN`.

Nonostante SQL non sia *case-sensitive*, è bene scrivere il costrutto e i parametri in maiuscolo, il resto della stringa in minuscolo.

```
SELECT nome,cognome,telefono
  FROM Atleti
  WHERE altezza>170 AND peso<95;
```

Attenzione al punto e virgola a fine comando.

MySQL connector

Affinché MySQL e Python possano interagire, dobbiamo installare un modulo tramite pip, come abbiamo imparato a fare in precedenza. Su Windows diamo:

```
pip install mysql-connector-python
```

Su Linux:

```
sudo apt install python3-mysql.connector
```

Creiamo un file test.py che contenga solo la riga:

```
1 import mysql.connector
```

Se il file viene eseguito senza errori, possiamo iniziare.

Controllo connessione

Avviamo XAMPP e il server MySQL. Creiamo un file come il seguente:

```
1  import mysql.connector
2  #connessione ad un database
3  mydb = mysql.connector.connect(
4      host = "localhost",  #per connetterci in locale
5      user = "root",       #utente principale
6      password = "",
7  )
8  print(mydb)
```

Se il programma ritorna una stringa come:

```
<mysql.connector.connection.MySQLConnection object at
↪  0x782cf375d7f0>
```

allora tutto funziona a dovere.

Creazione di un database

Per creare un nuovo database, dobbiamo utilizzare un cursore:

```
1  import mysql.connector
2
3  mydb = mysql.connector.connect(
4      host = "localhost",  #per connetterci in locale
5      user = "root",       #utente principale
6      password = "",       #nessuna password
7  )
8  #il cursore mi permette di ESEGUIRE COMANDI SQL in Python
9  mycursor = mydb.cursor()
10
11 mycursor.execute("CREATE DATABASE mydatabase")
```

▸ Documentazione sui cursori

Esercizi

1. Crea un database con il nome “Gatti”.
2. Scrivi un algoritmo che chieda all’utente il nome del nuovo database e crei poi un database con il nome inserito.

Elencare i database esistenti

Se vogliamo vedere tutti i database sul nostro sistema:

```
1  import mysql.connector
2
3  mydb = mysql.connector.connect(
4      host = "localhost",
5      user = "root",
6      password = "",
7  )
8
9  mycursor = mydb.cursor()
10
11 mycursor.execute("SHOW DATABASES")
12
13 for x in mycursor:
14   print(x)
```

Esercizi

3. Fai stampare a schermo l'elenco dei database presenti sul tuo computer (alcuni sono preimpostati in XAMPP) e controlla che il database creato nell'esercizio 1 esista.

Collegamento ad un database

In fase di connessione, possiamo specificare a quale database connetterci:

```
1  import mysql.connector
2
3  mydb = mysql.connector.connect(
4      host = "localhost",
5      user = "root",
6      password = "",
7      database="mydatabase",
8  )
9
10 ...
```

Eliminazione di un database

Se vogliamo eliminare un database, usiamo il comando **DROP**:

```
1  import mysql.connector
2
3  mydb = mysql.connector.connect(
4      host = "localhost", user = "root", password = "",
5  )
6
7  mycursor = mydb.cursor()
8
9  mycursor.execute("DROP DATABASE mydatabase")
```

Esercizi

4. Fai stampare a schermo l'elenco dei database presenti nel sistema, poi elimina il database dell'esercizio 1 e stampa la lista aggiornata.
5. Dopo aver ricreato i database "Gatti", "Cani" e "Pappagalli" , scrivi un algoritmo che mostri all'utente l'elenco dei database presenti nel sistema. Fai digitare all'utente il nome del database che vuole eliminare ed esegui l'eliminazione.
6. Scrivi un algoritmo che chieda all'utente quale tra le seguenti operazioni vuole compiere: mostrare i db, creare un nuovo db, eliminare un db esistente. In base alla scelta dell'utente, esegui l'operazione richiesta e, se necessario, mostra la lista dei db aggiornata.

Mostrare le tabelle

Per mostrare le tabelle di un database:

```
1  import mysql.connector
2
3  mydb = mysql.connector.connect(
4      host = "localhost", user = "root", password = "",
5      database="mydatabase",
6  )
7
8  mycursor = mydb.cursor()
9
10 mycursor.execute("SHOW TABLES")
11
12 for x in mycursor:
13   print(x)
```

Esercizi

7. Mostra le tabelle del database “mysql” preimpostato in XAMPP.

phpMyAdmin

phpMyAdmin è un client web open source per gestire database SQL di rete.

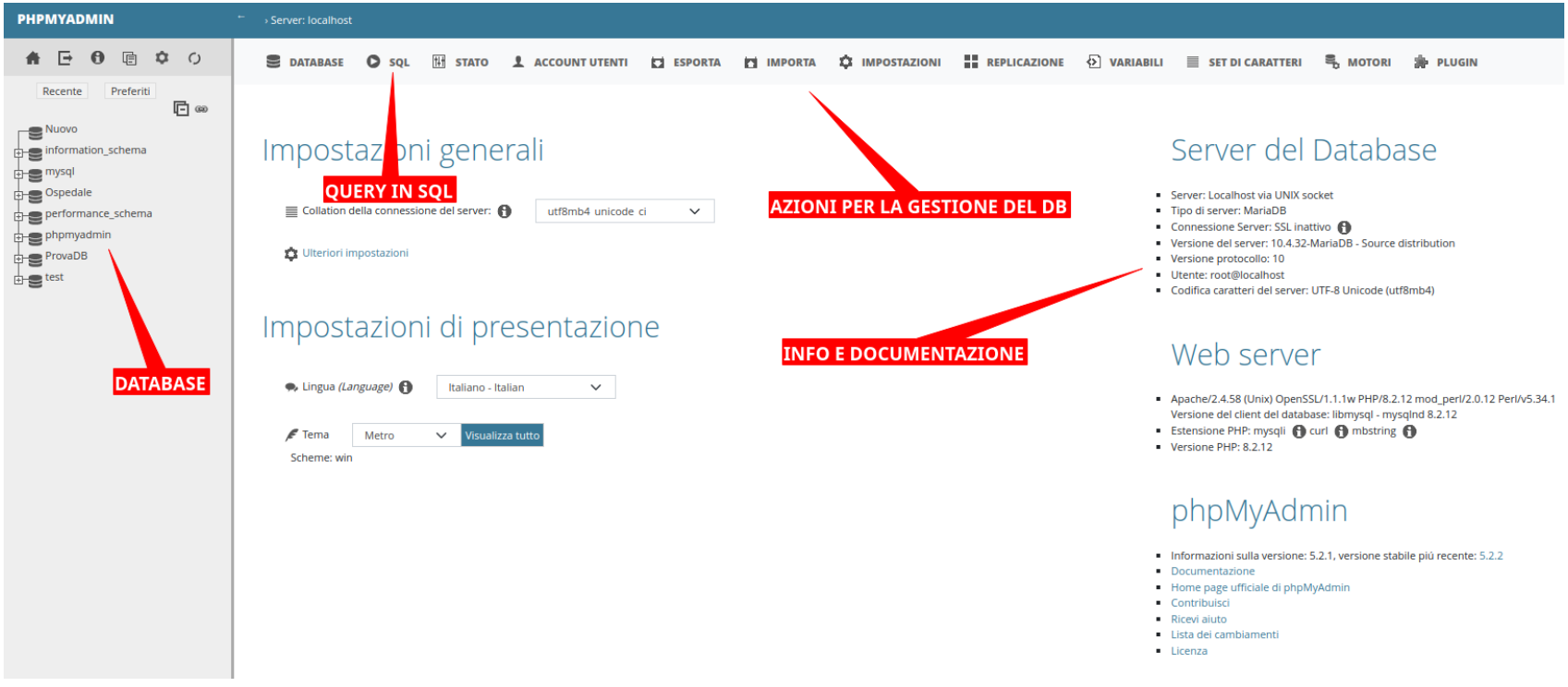
Possiamo eseguire operazioni sul database da riga di comando (come facciamo in Python) oppure via GUI.

Quando XAMPP è in esecuzione, il server locale può essere raggiunto dall'URL:

http://localhost/phpmyadmin/

oppure, su Windows, cliccando su "Admin" del modulo MySQL in XAMPP.

Interfaccia di phpMyAdmin



Gestione dei database con phpMyAdmin

Da phpMyAdmin possiamo vedere quali database abbiamo creato e quali dati vi abbiamo inserito.

Apriamo i nostri database dalla barra di sinistra. Possiamo:

- visualizzare e modificare i dati presenti nelle tabelle;
- eseguire comandi in linguaggio SQL;
- importare dati e interi database;
- visualizzare le relazioni (tab “Designer”);

e molto altro.

Importazione database di esempio

Prova a importare dall'interfaccia web di phpMyAdmin i database di esempio scaricati con il corso:

- db-pharma.sql;
- db-cosmesi.sql;
- db-profumi.sql.

Una guida all'importazione è contenuta nella cartella dei database di esempio.

Esercizi

Gli esercizi seguenti possono essere svolti solo dopo aver importato i database di esempio.

8. Scrivi un algoritmo che mostri le tabelle del database “db-cosmesi”.
9. Scrivi un algoritmo che mostri le tabelle del database “db-profumi”.
10. Scrivi un algoritmo che mostri le tabelle del database “db-pharma”.