Basi di Dati

CREAZIONE E POPOLAMENTO DI UNA BASE DI DATI

La finalità di questa esercitazione è quella di creare, date delle specifiche progettuale, appositi script di creazione e popolamento di una base di dati.

1. Passi preliminari per lo svolgimento dell'esercitazione

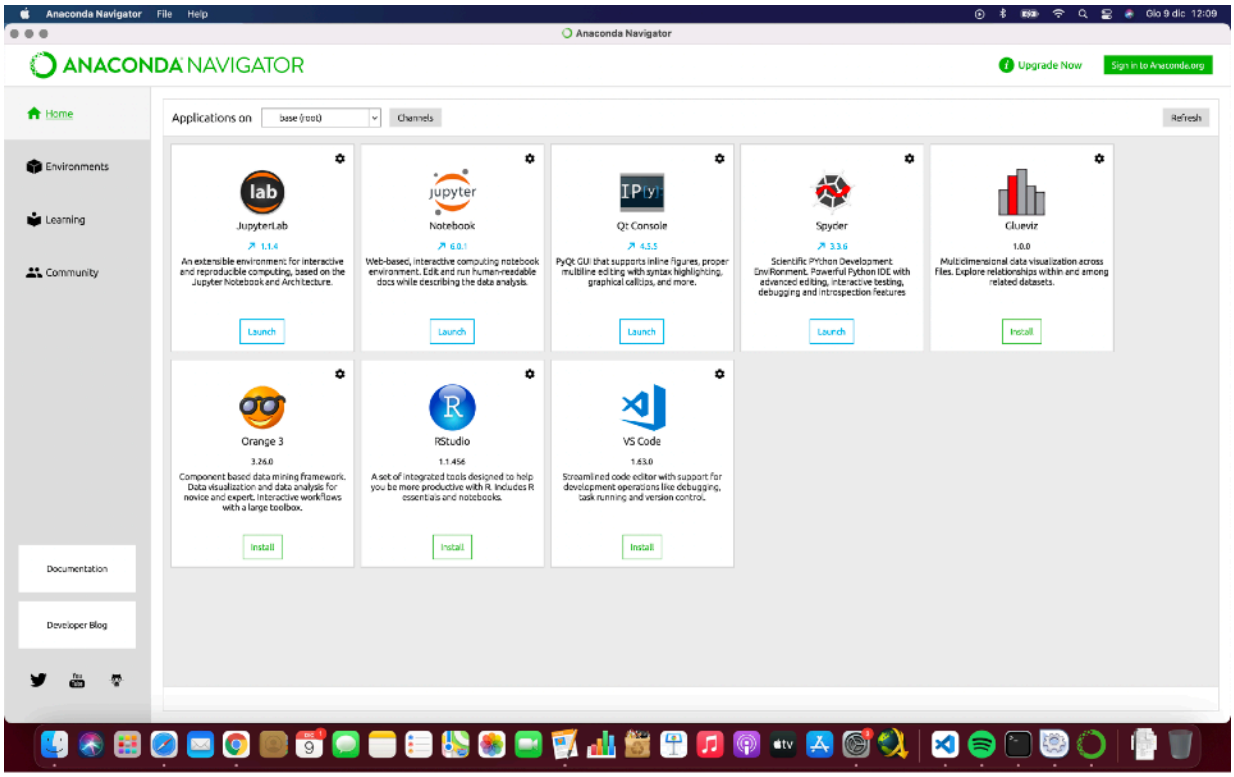
La finalità di questa seconda parte dell'esercitazione è di creare, dato lo schema logico di una base di dati, appositi script di creazione e popolamento della base di dati, e di scrivere ed eseguire alcuni comandi di aggiornamento e cancellazione utilizzando il linguaggio SQL.

Questa parte dell'esercitazione utilizza MySQL, e in particolare la versione disponibile nel prodotto XAMPP. Sarà inoltre necessario scaricare un driver per python per poter accedere ad un database MySQL.

Installare mysql.connector

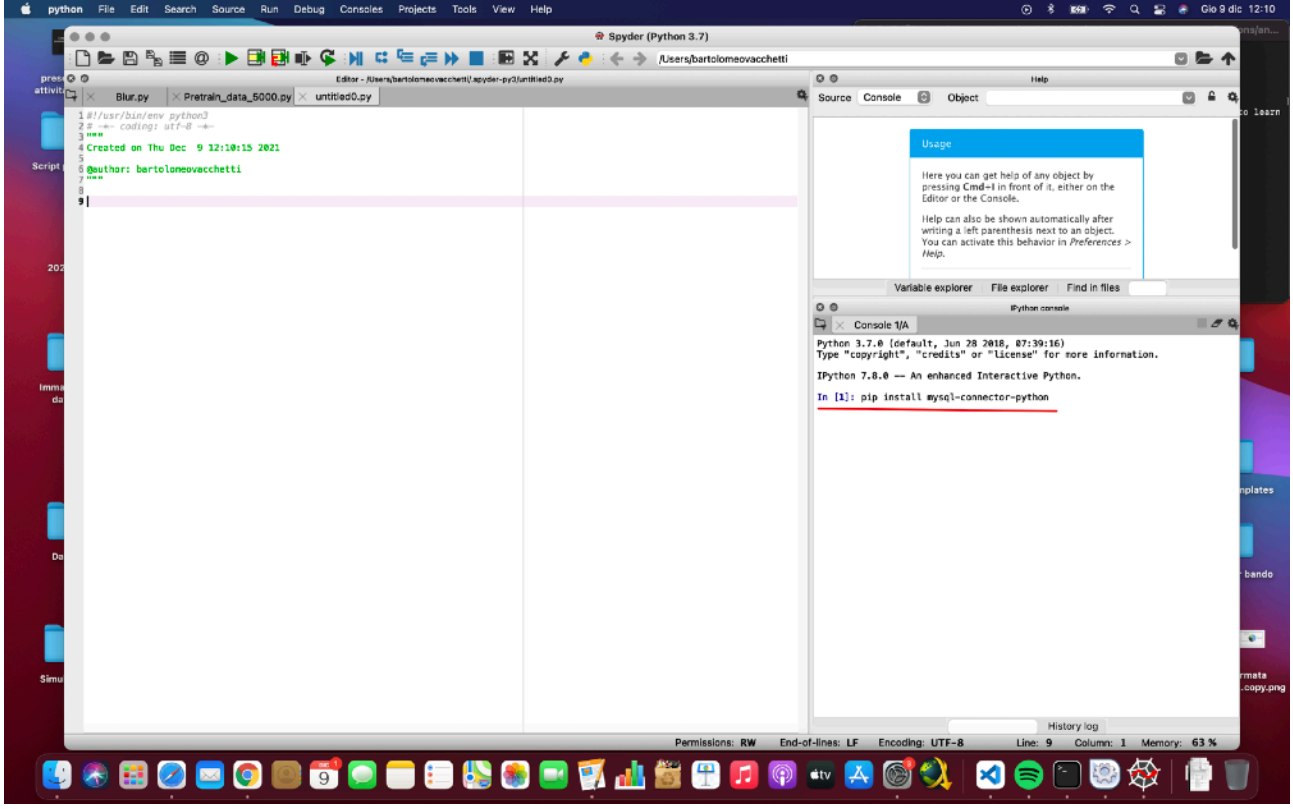
Per prima cosa occorre scaricare il driver *mysql.connector* per python. Per poterlo installare sui pc del lab accedere ad Anaconda Navigator.

Da Anaconda avviare spyder.



Una volta aperto spider utilizzare da terminale il comando

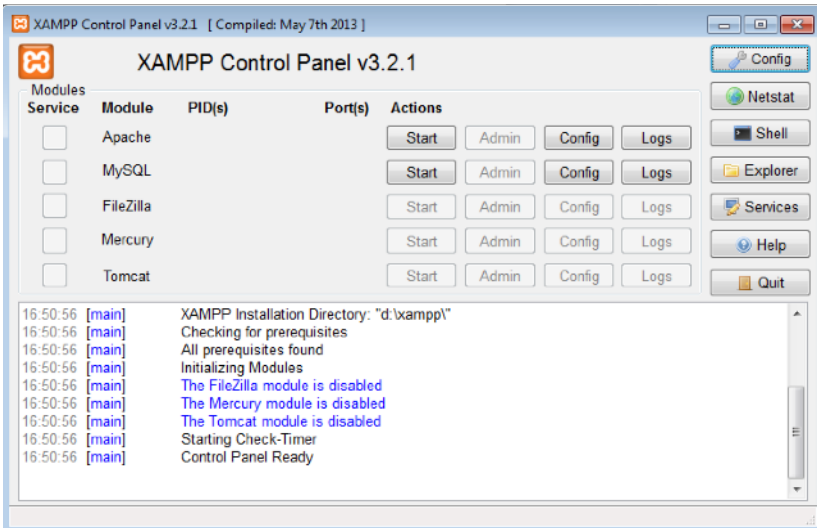
pip install mysql-connector-python



Avvio del server MySQL sulla macchina locale e avvio di Apache

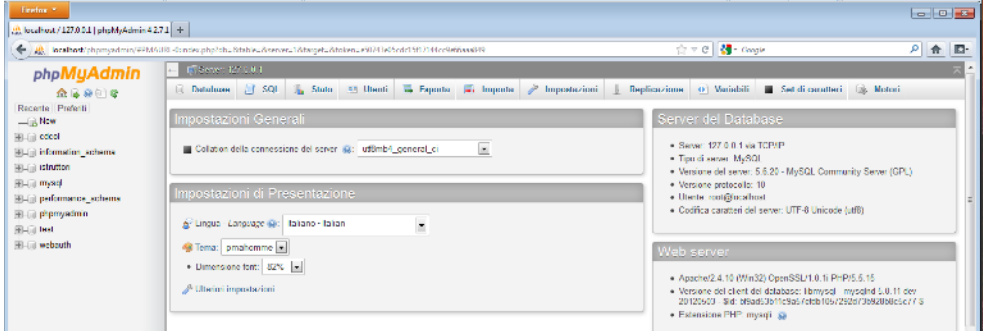
L'esecuzione degli script python contenenti i comandi SQL per la creazione e il popolamento della base di dati avviene tramite l'esecuzione dello script python. Per poter visualizzare meglio il database e le tabelle si usa l'interfaccia web di MySQL.
Prima di aprire l'interfaccia web di MySQL è necessario:

- Avviare il server locale Apache
- Avviare il server locale MySQL



In particolare, eseguire i seguenti passi:

1) Avviare il programma "XAMPP Control Panel"
2) Avviare Apache premendo il tasto Start nella riga relativa a Apache
3) Avviare MySQL premendo il tasto Start nella riga MySQL
4) Aprire l'interfaccia web di MySQL premendo il tasto Admin nella riga di MySQL (il browser si aprirà automaticamente sull'url associata alla pagina di amministrazione e interrogazione di MySQL)



5) La creazione di database e tabelle può essere eseguita tramite un script python. Dopo la corretta esecuzione dello script aggiornando phpMyAdmin compariranno database e tabelle. Sempre tramite script python si possono popolare le tabelle di dati e interagire con essi.Se le tabelle non compaiono dopo l'esecuzione dello script python bisogna aggiornare l'interfaccia (icona freccia verde in alto a sinistra sotto phpMyAdmin)
6) Per rilanciare più volte lo script di creazione/popolamento ricordarsi di cancellare eventuali istanze del database creato in precedenza dal panello Database oppure includere all'inizio dello script i comandi per la cancellazione delle tabelle preesistenti

1. Generazione degli script di creazione e popolamento del DB

1. Gli script sono semplici file di testo scritti con un qualsiasi editor (es., Notepad, Word, Wordpad). Tuttavia raccomandiamo di utilizzare editor appositi(PyCharm, Visual Studio Code, Spyder…)

2. Gli script vanno salvati con estensione *.py*

3. Gli script sono in python. Tuttavia si possono eseguire istruzioni SQL tramite il comando *execute()*.

4. Per interagire con il DBMS MySQL, sono necessarie le seguenti istruzioni preliminari (da scrivere all'inizio del file):

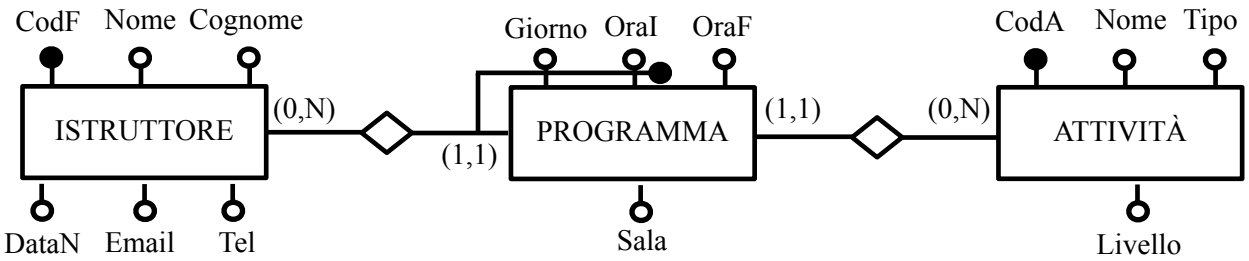
```
mydb = mysql.connector.connect(
  host='localhost',
  user="root",
  password="",
)

mycursor=mydb.cursor()

mycursor.execute("CREATE DATABASE IF NOT EXISTS palestra")
mycursor.execute("USE palestra")
```

5. Alle istruzioni preliminari seguono la sequenza di istruzioni in linguaggio SQL per la creazione e il popolamento del DB (CREATE TABLE e INSERT). Le istruzioni in sql andranno passate al comando *execute()* o *executemany()*. Se uno di questi comandi non vene eseguito non verranno eseguite le istruzioni in sql. Per popolare le tabelle ricordarsi di eseguire anche il comando *commit()*
6. Ricordarsi si verificare la sintassi e i tipi di dato compatibili con quelli richiesti dal DBMS MySQL.

2. Esercizi

Realizzare la base di dati corrispondente allo schema logico seguente relativo ad un alcune attività di una palestra (le chiavi primarie sono sottolineate e le chiavi esterne sono in corsivo).

Per ogni istruttore è noto il codice fiscale, il nome, il cognome, la data di nascita, l'indirizzo e-mail e il numero di telefono. Per ogni attività è noto il codice, il nome, il tipo (es. attività musicale) e il livello (un numero compreso tra 1 e 4). Il programma delle attività riporta il giorno (es. lunedì, martedì, ecc..) e l'ora di inizio e di fine in cui ogni istruttore svolge una determinata attività. Per ogni attività programmata è noto il numero della sala in cui si svolge.



ISTRUTTORE (CodF, Nome, Cognome, DataN, Email,

Tel) ATTIVITA' (CodA, Nome, Tipo, Livello)

PROGRAMMA (*CodF*, Giorno, OraI, OraF, CodA, Sala)

Svolgere le seguenti attività:

- Creare uno script python *creaDB.py* con le istruzioni per la creazione della base di dati corrispondente allo schema logico riportato(riscrivere istruzioni in python).
- Creare uno script SQL *popolaDB.sql* con le istruzioni per il popolamento della base di dati creata al punto precedente(scrrivere istruzioni python).

- Testare gli script di creazione e popolamento sul DBMS MySQL

In relazione alle attività precedenti, svolgere i seguenti passi:

- Specificare nello script di creazione del DB eventuali vincoli di dominio e/o di tupla appropriati e verificarne l'applicazione mediante l'interfaccia Web di MYSQL.

I vincoli presi in considerazione riguardano la presenza nella tabella Programma degli attributi CodA, identificatore univoco delle tabella AttivitàMusicale, e CodFisc, identificatore univoco delle tabella Istruttore.

```
mycursor.execute("CREATE TABLE IF NOT EXISTS Programma (CodFisc
VARCHAR(15) NOT NULL ,\
```

```
                Giorno VARCHAR(15) NOT NULL,\
                OraI TIME NOT NULL ,\
                OraF TIME NOT NULL ,\
                Sala
ENUM('1','2','3','4','5','6','7','8','9','10'),\
                CodA VARCHAR(255) NOT NULL ,\
                PRIMARY KEY(CodFisc,Giorno, OraI),\
                FOREIGN KEY(CodFisc)\
                    REFERENCES Istruttore(CodFisc)\
                    ON DELETE CASCADE\
                    ON UPDATE CASCADE,\
                FOREIGN KEY (CodA)\
                    REFERENCES Attivita(CodA)\
                    ON DELETE CASCADE\
                    ON UPDATE CASCADE)")
```

- Scegliere opportunamente le politiche di gestione dei vincoli più idonee al contesto analizzato.

Nel nostro caso come politica di gestione dei vincoli è stato scelto di usare l'opzione CASCADE, la quale propaga l'eventuale operazione di aggiornamento (ON UPDATE CASCADE) o cancellazione (ON DELETE CASCADE).

- Abilitare l'opzione di verifica automatica del vincolo di integrità referenziale (SET FOREIGN_KEY_CHECKS=1;) e verificare l'effetto su:
    - l'ordine delle istruzioni di creazione e popolamento delle tabelle

Usando il comando SET FOREIGN_KEY_CHECKS=1 (opzione di default su python); la verifica del vincolo di integrità viene eseguita automaticamente ad ogni istruzione, per tale motivo nella scrittura di uno script è necessario creare e popolare prima le tabelle referenzianti e poi quelle referenziate. In caso contrario si genera un errore. Il problema viene risolto disabilitando l'opzione di verifica automatica, (SET FOREIGN_KEY_CHECKS=0;). In questo modo l'ordine di creazione e popolamento delle tabelle nello script non è più rilevante, anche se potrebbe causare l'insorgere di inconsistenza.

    - presenza di eventuali inconsistenze nei dati

Se si usa SET FOREIGN_KEY_CHECKS=0; l'ordine di creazione e popolamento delle tabelle non è più rilevante, ma potrebbero sorgere delle inconsistenze in quanto i controlli dei vincoli di integrità non vengono effettuati automaticamente ad ogni istruzione. Quindi istruzione come la INSERT nella tabella referenziante e DELETE nella tabella referenziata potrebbero generare inconsistenze.

Esempio: "INSERT INTO Programma (CodFisc,Giorno,OraI,OraF,Sala,CodA)..". La insert prevede

l'inserimento di un valore nei campi CodFisc e CodA. Se le tabelle Istruttore e Attività non fossero state già create e popolate, i valori inseriti causerebbero problemi in quanto dati inconsistenti.

.