**Follosco - Comini**

**ESAME PYTHON**

**Documentazione di progetto**

**INDICE**

1. Introduzione
2. Script per la creazione degli Oggetti
3. Script principale per la creazione delle varie funzioni
   1. Funzione “def createDb”
   2. Funzione “def esegui”
   3. Funzione “def estrazione”
   4. Funzione “def id\_generator”
   5. Funzione “def add\_general”
   6. Funzione “def delete\_general”
   7. Funzione “def ricerca\_libro”
   8. Funzione “def ritardi”
   9. Funzione “def prestito”
4. Main Menu Principale
   1. Menu Autore
   2. Menu Categoria
   3. Menu Libro
   4. Menu Prestito
   5. Menu Ricerca
   6. Menu Utente
5. Codici SQL
   1. Schema
   2. DML
6. Extra
7. Conclusioni Personali

**1. Introduzione**

L’intento del presente progetto è la realizzazione di un’applicazione per la gestione di una biblioteca. Il software deve gestire per l’entità libro: la ricerca, il prestito, la restituzione, l’aggiunta, l’eliminazione e la modifica; per l’entità categoria: l’aggiunta e la cancellazione; per l’entità utente: l’aggiunta e la cancellazione. Dovrà anche gestire i prestiti in ritardo da parte del cliente e dovrà mostrare il catalogo dei libri presenti nel database.

La base di dati conterrà tutte le informazioni riguardanti i libri gestiti dalla biblioteca, gli utenti e i prestiti, oltre alle entità autore e categoria. Il linguaggio di programmazione per questo tipo di progetto è Python e SQLite il data base utilizzato. I dati sono stati gestiti con l’utilizzo delle classi per la parte di programmazione invece i dati persistenti sono stati gestiti, come già citato sopra, da un database.

**2. Script per la creazione degli Oggetti**

Al fine di ottenere una prima visione per la creazione della struttura della biblioteca abbiamo deciso di implementare questo progetto attraverso l’utilizzo degli oggetti in Python. Quest’ultimi ci permettono di raggruppare variabili e funzioni in maniera logica e riutilizzabile, consentendo di gestire progetti in modo semplice. Infatti abbiamo definito diverse classi, tra cui “class libro” e *“class persona”; quest’ultima* è la classe padre di “class utente” e “class autore” che sono le classi figlie. Tutte le classi saranno precedute dal metodo speciale “\_\_init\_\_” che consente di inizializzare gli attributi della classe.

Immagine che contiene testo

Descrizione generata automaticamente

*[Screen class persona, utente, autore]*

Immagine che contiene testo, interni, schermo, screenshot

Descrizione generata automaticamente

*[Screen class libro]*

**3. Script principale per la creazione delle varie funzioni**

Come prima cosa abbiamo importato gli script “add\_general” e “oggetti”. Successivamente abbiamo descritto il compito di ogni funzione presente nel progetto che verrà richiamata quando necessario.

Abbiamo documentato:

1. **La funzione “def createDb”** -> È la funzione che crea il data base su SQLite grazie ai seguenti script: ‘biblioteca.sql’, che crea lo schema; e ‘dml\_biblioteca.sql’ che popola il data base appena creato. In questa funzione è presente la ‘with’, grazie alla quale è possibile evitare di sovrascrivere il data base se questo esiste già. Infatti mi solleverà un’eccezione con scritto: “il database esiste, si suppone che esista anche lo schema”.

Immagine che contiene testo

Descrizione generata automaticamente

*[Screen connessione con il database SQLite]*

1. **La funzione “def esegui”** —> Tramite il parametro “conn” ci crea un cursore e ci consente di eseguire query verso il database. Per cautelarci da eventuali errori imprevisti, che potrebbero impedirci di chiudere la risorsa correttamente, usiamo invece il comando “with”. Come ultimo comando della funzione è presente il “fetchall ()” che ci restituisce tutte le righe come un elenco di tuple. Nel caso in cui non ci siano record da recuperare viene restituito un elenco vuoto.

Immagine che contiene testo

Descrizione generata automaticamente

*[Screen “def esegui”]*

1. **La funzione “def estrazione”** —> Estrae da una tabella del db, tutti gli elementi di una determinata colonna. Esempio: estrazione ("libro", "titolo")

Il primo parametro da passare è il nome della tabella, dove si inserisce il nome di una tabella dello schema del db. Il secondo parametro è la colonna, dove si inserisce il nome di una colonna della tabella seguente. Alla fine ritornerà zz che è una lista.

Immagine che contiene testo

Descrizione generata automaticamente

*[Screen “def estrazione”]*

1. **La funzione “def id\_generator”** —> Genera una chiave id per le classi utente, libro, autore e categoria. La nuova chiave si ottiene sommando la chiave massima di 1. I parametri passati sono: classe utente, classe str, classe Libro o classe autore. Ritornerà come risultato un idd di tipo intero.

Immagine che contiene testo

Descrizione generata automaticamente

*[Screen “def id\_generator”]*

1. **La funzione “def add\_general”** —> Permetterà di inserire l’utente, la categoria, il libro e l’autore attraverso le query in SQL, senza creare funzioni inutili in aggiunta. All’interno della funzione è presente la funzione “id\_generator” che creerà i vari identificatori degli oggetti. La funzione “add\_general” esegue le query sql di INSERT INTO per aggiungere un oggetto nel db che può essere libro, utente, categoria, autore. I parametri da passare sono conn, cioè la connessione e l’oggetto scelto. Questa funzione restituirà la prima key generata di tipo intero.

Le query scritte per i vari inserimenti sono:

Per l’inserimento dell’utente: *'''INSERT INTO utente (id\_tessera, data\_registrazione, nome, cognome, telefono, indirizzo, email) values (?, ?, ?, ?, ?, ?, ?)'''*

Per l'inserimento della categoria: *’''INSERT INTO categoria (id, nome) values (?, ?)'''*

Per l'inserimento del libro*: ’''INSERT INTO libro (isbn, titolo, lingua, editore, anno, copie) values (?, ?, ?, ?, ?, ?)'''*

Per l'inserimento dell'autore del libro: *’''INSERT INTO autore (id, nome, cognome, data\_nascita, luogo\_nascita, note) values (?, ?, ?, ?, ?, ?)'''*

Di seguito sono state implementate ad oggetti le classi: utente, categoria, libro e autore con i relativi controlli. Gli oggetti categoria e autore verranno inizializzati, a meno che il nome di questi non sia già presente nel database.

1. **La funzione “def delete\_general”** —> Consente di cancellare:

* la categoria solo se non è associata a più libri;
* l’utente se non ha libri in prestito, quindi non restituiti;
* il libro solo se non è in prestito da un utente.

In questa funzione sono presenti anche le tabelle “bridge\_categoria”, “bridge\_autore” e “prestito” che permetteranno di cancellare un libro in cascata attraverso l’ISBN scelto.

Immagine che contiene testo

Descrizione generata automaticamente*[Screen eliminazione in cascata del libro]*

Delete\_general esegue le query sql di DELETE per rimuovere una riga nel db. I parametri che abbiamo passato sono: la connessione, la riga da cancellare e il numero intero associato alla tabella da cancellare del db (0 per cancellare nella tabella categoria, 1 per cancellare nella tabella utente, 2 per cancellare nella tabella libro, 3 per cancellare nella tabella autore).

1. **La funzione “def ricerca\_libro”** —> Ricerca il libro (tramite ISBN) all'interno del database e restituisce un oggetto libro. I parametri passati sono la connessione al db e il numero dell’ISBN da ricercare, questi consentiranno di ottenere come variabile di ritorno un libro che è contenuto nella classe libro.

La query è scritta così come riportato: ’

*’'SELECT libro.isbn, libro.titolo, libro.lingua, libro.editore, libro.anno, libro.copie, group\_concat(distinct categoria.nome), group\_concat(distinct autore.cognome)*

*FROM libro*

*join bridge\_categoria ON libro.isbn = bridge\_categoria.isbn\_libro*

*join categoria ON categoria.id = bridge\_categoria.id\_categoria*

*join bridge\_autore ON libro.isbn = bridge\_autore.isbn\_libro*

*join autore ON autore.id = bridge\_autore.id\_autore*

*WHERE libro.isbn = :filtro*

*group by libro.isbn '''*

1. **La funzione “def ritardi”** —> Cerca nel db i libri presi in prestito dall'utente che non sono stati restituiti entro la data di scadenza. I parametri passati sono la connessione e il numero di tessera dell’utente. La funzione fornisce in uscita una lista dei libri che ancora non sono stati consegnati entro la data di scadenza.

La query scritta è:

*'SELECT isbn\_libro FROM prestito WHERE data\_prestito = :date'*

1. **La funzione “def prestito”** —> Inserisce nel db il libro e l'utente che lo ha preso in prestito e quindi calcola la data di prestito e di restituzione del libro (max 30 giorni) e aggiorna il numero di copie dei libri presi in prestito. I parametri passati, oltre alla connessione, sono, come già citato sopra, il libro e il numero di tessera dell’utente.

Le query scritte sono:

*'''INSERT INTO prestito (isbn\_libro, tessera\_id, data\_prestito, data\_restituzione) values(?, ?, ?, NULL)''*

*'''UPDATE libro SET copie = :n\_copie WHERE isbn = :filtro'''*

1. La funzione “def update\_libroDB” -> Permette di aggiornare un campo della tabella libro che può essere: lingua, editore, anno o copie. I parametri passati sono, dopo la connessione al data base, l’ISBN del libro scelto, il campo designato per il cambio e il nuovo valore da inserire. La query che permette questo update di dati è:

*'''UPDATE libro SET {campo} = :valore WHERE isbn = :filtro’''*

1. **La funzione “def restituzione”** --> Con la query SQL è stato settato, in un primo momento, il numero di copie dei libri presenti nel db e, in un secondo momento, è stata settata la data del prestito del libro messa poi a confronto con la data di riconsegna l’ISBN del libro e il numero di tessera dell’utente.

Le query scritte per questa funzione sono:

*'''UPDATE libro SET copie = :n\_copie WHERE isbn = :filtro’''*

*'''UPDATE prestito SET data\_restituzione = :data*

*WHERE isbn\_libro = :filtro AND tessera\_id = :filtro2 ‘''*

1. La funzione “def ricerca\_prestito” ->

Grazie alla query:

*'''SELECT tessera\_id, sbn\_libro FROM prestito WHERE data\_restituzione is NULL AND tessera\_id = :tessera’''*

È possibile ricercare tutti i prestiti attivi per un utente della biblioteca.

**4. Main Menu Principale**

Come terzo script in Python è presente il Main dove vengono importati da subito gli altri script, ossia: query\_sql, MenuAutore, MenuCategoria, MenuLibro, MenuLibro, MenuPrestito, MenuRicerca e MenuUtente. In questo script l’utilizzatore del programma ha la possibilità di effettuare le chiamate al database a seconda delle iterazioni che vuole svolgere, selezionandole dal Menu Principale. Infine, sempre in questo script, è possibile trovare la funzione catalogo che mostra tutti i libri presenti nel database.

1. **Menu Autore**

Con questo script, sottomenu del main, è possibile inserire o cancellare un autore.

La creazione di un autore avviene grazie alla funzione “def crea\_autore” dove si potrà inserire un nuovo autore nel caso in cui esso non fosse presente già nel database.

L’eliminazione di un autore avviene grazie alla funzione “def DeleteUtente” che andrà a verificare se esso è presente nel database e di seguito lo cancellerà.

1. **Menu Categoria**

Con questo script, sottomenu del main, è possibile inserire o cancellare una categoria. Infatti, sono presenti due funzioni: una che aggiunge una categoria, “def AddCategoria”, se non è già presente nel database; e una che elimina una categoria, “def DeleteCategoria”, se è già inserita nel databse.

1. **Menu Libro**

Con questo script, sottomenu del main, è possibile inserire, cancellare o aggiornare un libro.

Le funzioni create sono:

“def create\_libro” - > Permette di creare un nuovo libro con vari dati da inserire. È importante ricordarsi di inserire un ISBN di 13 cifre intere altrimenti, il programma, solleverà un’eccezione. Sono presenti anche due cicli che permettono l’inserimento di più categorie e autori.

“def AddLibro” -> Permette di controllare se il libro che si vuole inserire è già presente nel data base.

“def DeleteLibro” -> Permette di eliminare un libro se esso è presente nel database. Ricordarsi di inserire in modo corretto l’ISBN altrimenti solleverà un’eccezione.

“def UpdateLibro” - > Permette di aggiornare un libro presente nel data base. In particolare, consente di cambiare: la lingua, l’editore e l’anno in cui è stato scritto il libro e il numero di copie presenti nel database del libro.

1. **Menu Prestito**

Con questo script, sottomenu del main, è possibile prendere in prestito un libro dalla biblioteca o restituire un libro già preso in prestito. Per prima cosa bisogna, grazie alla funzione “def PrestaLibro”, controllare se il libro scelto è presente nel data base della biblioteca; come seconda cosa è necessario controllare se l’utente ha già dei libri presi in prestito.

Per restituire un libro ci serviamo, invece, della funzione “def RestituzioneLibro” che va a controllare se il libro che si vuole restituire è presente nella tabella dei libri restituiti e, in un secondo momento, richiede il numero di tessera dell’utente che intende restituire il libro.

1. **Menu Ricerca**

Con questo script, sottomenu del main, è possibile ricercare un libro, ricercare un determinato prestito di un libro e ricercare se il libro è in ritardo per la restituzione. Questo script è diviso in tre funzioni: la funzione “def Ricercalibro” che controlla se il libro da ricercare, tramite l’ISBN, è presente nel data base; la funzione “def RicercaPrestiti”, che controlla se l’utente ha ottenuto in prestito qualche libro; infine la funzione “def RicercaRitardi”, che controlla se l’utente ha prestiti in ritardo.

1. **Menu Utente**

Con questo script, sottomenu del main, è possibile inserire, cancellare un utente. Sono presenti tre funzioni: “def crea\_utente”, “def AddUtente” e “def DeleteUtente”. La prima permette di creare un nuovo utente per il database biblioteca; la seconda di aggiungere il numero di tessera al nuovo utente; la terza di cancellare un utente se esso è presente nel data base

**5. Codici SQL**

Infine, come script SQL, sono stati creati lo schema e il dml che verranno poi richiamati nel main.

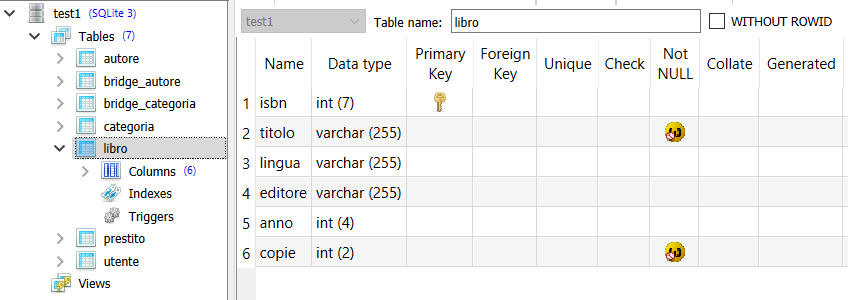
1. **Schema**

Il file, con all’interno lo schema del database, crea le tabelle attraverso la funzione: *CREATE TABLE nome\_tabella (nomi delle colonne della tabella) e le inserisce nel db SQLite*.

Immagine che contiene testo

Descrizione generata automaticamente

*[Screen esempio creazione di una tabella in Python]*

**

*[Screen esempio creazione di una tabella su SQLite (cambiare immagine)]*

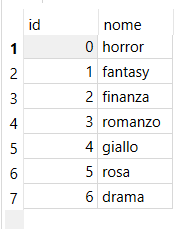
1. **DML**

Invece il file con all’interno il dml, attraverso la funzione: *INSERT INTO nome\_tabella (nomi delle colonne della tabella) VALUES (valori presenti nelle colonne della tabella)*, popolerà il database biblioteca con i primi dati inseriti dal programmatore.

Immagine che contiene testo

Descrizione generata automaticamente

*[Screen esempio di un’inserimento ‘insert into’ in Python]*

**

*[Screen esempio di dati su SQLite]*

**6. Extra**

Per facilitare la stesura del codice abbiamo importato i seguenti moduli:

Import datetime

Import sqlite3

Import os

Import sys

Per avere una condivisione più efficiente di tutti i file è stato usato GitHub.

Link di GitHub: <https://github.com/binod12345678/biblioteca>

**7. Conclusioni Personali**

Grazie a questo progetto, abbiamo potuto lavorare di squadra come se fossimo

realmente dei programmatori in un ambito lavorativo.

Infatti, ci siamo scambiati le risorse con il software Github e abbiamo imparato

a stilare una relazione descrittiva del nostro progetto.

Ringraziamo il docente per averci fatto imparare il linguaggio

di programmazione (Python) e speriamo di ritrovarci in un’altra unità formativa.

Grazie, Follosco & Comini