Relazione ufficiale programma Database scuole

Made by Andrea Manzo, Samuel Colombo and Lorenzo Cavasino

Indice

Introduzione:

Ci è stato assegnato il compito di scrivere un codice python che fosse in grado di convertire la tabella di un file XLS in un database in SQLite3.
Questa relazione fornisce una panoramica tecnica della soluzione implementata per la trasformazione di dati da un file XLS a un file JSON, per poi trasferirli in un database SQLite3. Abbiamo diviso il lavoro in due parti principali:

- Da XLS a JSON
- Da JSON a SQLite3

Da XLS a JSON:

Il primo passo è stato quello di scrivere il codice per leggere nel modo corretto i dati dal file XLS e tradurli in formato JSON. Dopo aver avuto una base di partenza da internet abbiamo modificato il codice affinché il formato finale rappresentasse tutti i dati nel modo corretto. I problemi principali incontrati in questa parte era uno, cioè non doveva leggere le prime 3 righe dell'excel che contenevano dati inutili e dalla riga 27 in poi, ho risolto il problema usando il ciclo sottostante:

```
for row_index in range(4, min(27, sheet.nrows)):  # Ignora le righe dalla 27 in poi
    row_data = sheet.row_values(row_index)
    # Ignora le colonne "Z" e "AA"
    row_data = row_data[:25] + row_data[27:]
    # Ignora le colonne da "C" a "J"
    row_data = row_data[:2] + row_data[10:]
    # Leggi le prime due celle normalmente
    first_cells = row_data[:2]

    data.append({columns[i]: row_data[i] if i < len(row_data) else None for i in range(len(columns))})
```

Le parti principali di questa parte di programma erano queste:

- Lettore di file Excel (xlrd): Il codice Python utilizza la libreria xlrd per aprire il file Excel specificato e leggere i dati dal primo foglio di lavoro.
- Trasformazione dei Dati: I dati letti dal file Excel vengono trasformati in una struttura di dati Python e successivamente serializzati in formato JSON.
- Serializzatore JSON: Utilizzato per convertire i dati estratti dal file Excel in un formato JSON.
- Aggiunta dei dati al file JSON: dopo aver estratto tutti i dati dall'Excel, abbiamo prima verificato se ci fosse o no un file JSON contenente dati di altre classi; nel caso in cui questo file non esista allora procediamo alla creazione di esso, altrimenti apriamo il file in questione ib lettura-scrittura e aggiungiamo i dati della nuova classe in append al file.

Da JSON a SQLite3:

Il secondo passo è stato quello di scrivere il codice per trasformare i dati dal formato JSON al database SQLite3. Per svolgere questo passaggio, dato che abbiamo considerato il fatto che due classi possano avere materie differenti, abbiamo fatto in modo che il programma analizzasse i dati nel JSON e creasse un database a doc, con tutte le materie di tutti gli studenti, mettendo a null i campi relativi ad una materia non avuta da uno studente.
Per svolgere questo passaggio abbiamo riscontrato un problema: per selezionare i campi da cui costruire il database abbiamo messo in una lista le varie materie prese dal JSON, tuttavia quest'ultime venivano messe nella lista in modo disordinato,e perciò prima di costruire il database abbiamo dovuto riordinarle manualmente.

Gli step principali sono stati:

- Gestore di database SQLite3: Utilizzato per creare e gestire un database SQLite3 e per persistere i dati estratti dal file Excel.
- Ricostruzione del database: una volta aggiornato il file JSON il database veniva cancellato e poi ricostruito sulla base dei nuovi dati inseriti nel file JSON.
- Controllo di Integrità: Il sistema controlla se il file JSON è stato correttamente generato e se il database SQLite3 è stato correttamente aggiornato.

Struttura del database

La struttura del database non è una struttura statica, in quanto esso cambia a seconda delle materie inserite nell'Excel, tuttavia ci sono degli attributi costanti, che sono:

- Nella prima colonna c'è il dato "Pr_" (numero di registro).
- Nella seconda colonna c'è il dato "Alunno" (nome e cognome dell'alunno).
- Nella terza colonna c'è il dato "classe" (nel formato *AnnoSezioneIndirizzo*)
- Successivamente ci sono tutte le materie che variano, nel caso in cui un alunno non abbia voti nella materia (in quanto vengono inserite le materie di tutte le classi e non tutte le classi hanno le stesse materie), questo dato verrà inserito come *NULL*.
- Nella penultima colonna c'è il dato "media".
- Nell'ultima colonna c'è il dato "esito" (se l'alunno è stato ammesso, rimandato o non ammesso all'anno successivo).

Pr_	Alunno	classe	ARTE	CHIMICA	COMPORTAMENTO	DIRITTO_ED_ECONOMIA	EDUCAZIONE_CIVICA	FISICA	GINNASTICA	KARATE	LINGUA_E_LETT_IT	LINGUA_INGLESE	MATEMATICA	RELIGIONE	SC_DELLA_TERRA_GEO	STORIA	TELECOMUNICAZIONI	Tecn_e_Tecn_di_rappr	Tecn_informatiche	Media	Esito
23	23 23	1Sim	NULL	6	9	7	8	6	8	NULL	6	7	6		6	6	NULL	7	8	6.92	Ammesso/a
1	1 1	1Bit	NULL	9	10	9	9	8	NULL	NULL	8	7	10	O	9	8	7	8	9	8.54	Ammesso/a

Eseguibile di avvio

Per garantire un corretto funzionamento del programma abbiamo anche programmato un programma in c, che si presenta come un eseguibile che, innanzitutto controlla che nell'ambiente d'esecuzione del programma siano installate tutte le librerie necessarie all'esecuzione di questo.
Nel caso in cui esse non siano installate verranno installate automaticamente dall'eseguibile.
Successivamente il programma richiederà di inserire la lista dei file da analizzare andando a controllare che siano file Excel e che siano presenti nella cartella del programma.
Nel caso in cui si vuole eseguire il programma senza farlo dal .exe bisognerà eseguirlo tramite riga di comando, aggiungendo dopo il nome del programma i nomi dei file da analizzare (in questo modo: *python programma.py file1.xls file2.xls file3.xls*).

Installazione del programma

Il programma può essere comodamente scaricato dal git pubblico pubblicato su Github al seguente indirizzo.
Nel caso in cui il programma o l'eseguibile abbiano dei malfunzionamenti si provi inizialmente ad reinstallarlo, altrimenti ci contatti a uno sei seguenti indirizzi:

- Andrea Manzo.
- Samuel Colombo.
- Lorenzo Cavasino.

Future implementazioni

Una futura implementazione potrebbe essere quella che, quando viene inserita una classe che è già stata precedentemente inserita nel database il programma sia in grado di riconoscerla e la scarti.

Conclusione:

La soluzione proposta offre un modo efficiente e automatizzato per importare e persistere i dati da un file XLS a un database SQLite3. Questo rende più semplice e veloce l'analisi e la gestione dei dati, consentendo agli utenti di concentrarsi sulle attività di analisi piuttosto che sul processo di importazione dei dati.