Progetto Applicazioni Web

# Studenti

Rosilde Garavoglia, matricola: 20034672

Roberto Aitchison, matricola: 20035054

# Istruzioni per l’utilizzo

L’applicazione web viene eseguita sulla porta 7070.

Nella schermata iniziale si chiederà di aprire il database del tipo desiderato. In particolare, le opzioni sono tre:

* Aprire un database di tipo SQLite, inserendone il path
* Aprire un database di tipo SQLServer, inserendo il nome del server, il nome del database, lo username e la password
* Aprire un database di tipo SQLServer con trusted connection, inserendo il nome del server e il nome del database (connection string con trusted\_connection = true)

Una volta connessi al database, si potranno visualizzare tutte le informazioni ad esso relative. Sarà possibile anche chiudere la connessione col database, aprire un’altra connessione, aprire una o più finestre che permettono di eseguire query SQL, esportare in csv o importare da csv.

Nel caso in cui siano aperte più connessioni dello stesso tipo di database, le operazioni di query, esportazione in csv e importazione da csv saranno effettuate sull’ultimo database aperto. Per eseguirle su un altro database precedentemente selezionato basterà chiudere le connessioni ad esso successivamente aperte.

Per l’operazione di esportazione in csv, digitare nell’apposito spazio il path dove si vuole che tutti i file vengano generati (per esempio una directory). Viceversa, per quanto riguarda l’importazione da csv, digitare nell’apposito spazio il path del file csv dal quale importare i dati.

# Scelte implementative

## Conservazione delle connection string

Per poter aprire più database dello stesso tipo, i due context (uno per tipo di DBMS) fanno riferimento alla corrispondente classe statica contenente una lista di stringhe, contenente tutte le connection string di quel tipo di database inserite dall’utente. Essi si connetteranno utilizzando l’ultima connection string della lista.

Quando una connessione viene aperta, la connection string verrà inserita in coda alla lista, così che il context faccia ora riferimento al database ad essa corrispondente. Qualora invece la connessione dovesse essere chiusa, la connection string verrà rimossa dalla lista, eliminandone l’ultimo elemento.

## Ottenimento dei metadati del db

Per l’ottenimento dei metadati del db, vengono innanzitutto estratti i nomi di tutte le tabelle tramite le query:

* SELECT name FROM sqlite\_schema WHERE type='table' per SQLite
* SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES per SQLServer

Questi vengono utilizzati per ottenere il nome, il tipo di dato e se il valore possa essere **NULL** di ogni colonna delle tabelle, tramite le query:

* PRAGMA table\_info({tabella}) per SQLite
* SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = '{tabella}' per SQLServer

Per ottenere le primary keys vengono utilizzate le query:

* PRAGMA table\_info({tabella}) per SQLite
* EXEC sp\_pkeys '{tabella}' per SQLServer

Infine, sempre tramite il nome della tabella, vengono prese le foreign keys, tramite le query:

* PRAGMA foreign\_key\_list({tabella}) per SQLite
* EXEC sp\_fkeys '{tabella}' per SQLServer

E gli indici definiti sulla tabella, tramite le query:

* PRAGMA index\_list({tabella}) per SQLite
* EXEC sp\_helpindex '{tabella}' per SQLServer

Una volta ottenuti questi dati, l’interfaccia presenta all’utente tutte le tabelle del db e le loro informazioni.

## Esecuzione di query sul db

L’esecuzione delle query viene svolta tramite l’inserimento della query che si desidera effettuare nell’apposito riquadro dell’interfaccia. Il codice analizza il tipo di query inserita e la esegue sul database.

Tramite le query:

* SELECT changes() per SQLite
* SELECT @@ROWCOUNT per SQLServer

È possibile ottenere il numero di righe del db che sono state interpellate nell’ultima query eseguita, tale numero viene stampato a schermo insieme alla tabella modificata.

## Importazione ed esportazione con csv

Per quanto riguarda l’esportazione, vengono per prima cosa ottenuti il nome delle tabelle, per ciascuna tabella il nome delle colonne e tutti i dati presenti nel database. Per fare questo vengono utilizzate le seguenti query:

* Query per SQLite
* Per ottenere il nome di tutte le tabelle: SELECT name FROM sqlite\_schema WHERE type='table'
* Per ottenere, per ogni tabella, il nome delle sue colonne: PRAGMA table\_info({tabella})
* Per ottenere tutti i dati di una tabella: SELECT \* FROM {tabella}
* Query per SQLServer
  + Per ottenere il nome di tutte le tabelle: SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES
  + Per ottenere, per ogni tabella, il nome delle sue colonne: SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = '{tabella}'
  + Per ottenere tutti i dati di una tabella: SELECT \* FROM {tabella}

Una volta ottenute tutte queste informazioni, si va a comporre il file (con nome {tabella}.csv, inserito nel path indicato dall’utente) per ciascuna tabella inserendo nella prima riga i nomi delle colonne e nelle righe successive i dati. Se non è presente esso viene creato, altrimenti viene sovrascritto.

L’importazione, invece, legge da file csv (indicato nel path inserito dall’utente) i dati necessari (il nome della tabella ricavato dal nome del file, le colonne ricavate dalla prima riga e i dati ricavati dalle righe successive). Una volta ricavati, essi vengono inseriti nel database con la query (ripetuta per ogni riga di valori del file):

* INSERT INTO {tabella} ({colonne}) VALUES({valori}) (valida sia per SQLite che per SQLServer)