CSE 310 – SQL Database Workshop

Example Classroom Code

- Starting Code: https://replit.com/@cmacbeth/CSE310SQLDBWorkshop
- Solution Code: https://replit.com/@cmacbeth/CSE310SQLDBWorkshopSolution

Useful Reference Links

- https://docs.python.org/3/library/sqlite3.html
- https://www.sqlite.org/lang.html
- https://www.w3schools.com/sql

Common SQL Commands using SQLite Syntax

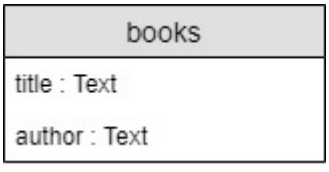
```
CREATE TABLE IF NOT EXISTS table (column1 TEXT, column2 INT, column3 REAL, …)

SELECT columns FROM table WHERE condition ORDER BY column ASC|DESC

INSERT INTO table VALUES (value1, value2, …)

UPDATE table SET column = value WHERE condition

DELETE FROM table WHERE condition
```

Python sqlite3 Library

A SQLite database is a binary file in your local directory. To open the file, use the `connect` function. The `cursor` is a reference to results from commands that we give to the database.

```
connection = sqlite3.connect('database.db')
cursor = connection.cursor()
```

The python library will execute SQL commands (strings) using the `cursor` object and the `execute` function. A common first command you will always need is to create the database table. In these examples, a simple books table is created.




```
cursor.execute("CREATE TABLE IF NOT EXISTS books (title TEXT, author TEXT)")
```

When you query the database, the `cursor` will provide the results via an iterator by calling the `fetchall` function.

```
cursor.execute("SELECT * FROM books ORDER BY title")
for book in cursor.fetchall():
    print(book[0]) # Assumes title is the first column in the table
```

When doing an SQL command to change the database such as INSERT, UPDATE, or DELETE, the database will change but that change will not be available to other users connected to the database until it is committed. The `commit` function should be called after modifying the database. If you close the database without committing the changes, then the changes will be lost.

```
cursor.execute("INSERT INTO books VALUES ('War and Peace', 'Tolstoy')")
connection.commit()
```

The `cursor` can also be used to check how many rows were modified.

```
cursor.execute("DELETE FROM books WHERE title = 'Paradise Lost'")
connection.commit()
if cursor.rowcount == 0:
    print("Book does not exist")
```

The `execute` function can also take a parameterized list of values to create the SQl command. This can be useful when you are have variables in your code. Question marks are used as placeholders.

```
def update_author(cursor, title, author):
    values = (title, author) # Order is important here
    cursor.exeute("UPDATE books SET author = ? WHERE title = ?", values)
    cursor.connection.commit() # You can get the connection from the cursor
```