Homework 7

Database-Backed Applications

In this exercise, you will write two programs that "talk" to a relational SQLite database. The first program will "seed" some data in the database. The second program is for an end user, and will present a simple text-based interface to enable the user to see the data in the database, as well as store new data in the database.

In the end, you will experience a programming idiom that will serve you well into the future: the steps that all programs execute when communicating with a relational database management system. Once you learn this idiom, you will find that it is the same for just about every programming language out there, and for every database-backed application.

Seeding the Database

Given the provided relational database about superheroes, write a program in any language that you wish that stores some data in the database and establishes some relationships. This program should:

- be named **seed.*ext***, where *ext* is the typical extension for your programming language
- insert three `villains` records
- insert three `teams` records
- insert seven `powers` records
- insert seven `colors` records
- insert three `superheroes` records
- establish relationships between superheroes and villains, teams, colors and powers

Once you get the program working, delete all the records in your database and run the program once, such that your database is seeded with some data.

A User-Facing, Database-Backed Application

Now, write a second program. This program is user-facing, and should provide the user

with a simple text-based interface for displaying and inserting data into the database. For example, when running your program, the user might see:

```
Welcome to the superheroes archive!
1. Superheroes
2. Villains
3. Teams
4. Powers
5. Colors
Your command: __
```

When a user enters any of the above numbers, the program should list all the records in each respective table. In the case of superheroes, each superhero should be represented as:

```
Miss Magic
Team: Supercrew
Nemesis: Mr. Evil
Powers: Flying, Crushing, Snapping you like a twig
Costume colors: black, blue
```

Once the data is displayed, your program should enable the user to create a new record or exit the program. For example, after the user selects 5 for colors, we might see:

```
Costume Colors
black
blue
pink
red
brown
purple
(A)dd a new color, or (Q)uit ?
```

Selecting Q would quit the program, but selecting A would then prompt the user to enter the attributes for the particular entity. For example, in the case of adding a new color, we might see:

```
Adding a New Color
Color Name: ____
```

Which waits for the user to enter a color name. Your program should then store the value in the respective table and then terminate.

When adding a superhero, the program should present a numerically-labeled list of options for each associated attribute. You can just use the primary keys for these numeric labels. For example,

```
Adding a New Superhero
Name:___
Arch enemy:
1. Dr. Octagon
2. Black Star
3. Miss Chaos
>_
Team:
1. Bad girls
2. Bad boys
>_
Colors:
1. Red
2. Blue
>_
Powers:
1. X-Ray Vision
2. Superhuman strength
```

```
3. Eating a dozen donuts in one swallow
>_
```

In the case of colors and powers, the user should be able to specify multiple values. How you handle this is up to you.

Remember, in every case, after adding a new record your program can then just terminate.

Additional Challenges

Add additional features that let the user delete any record.

Add additional features that let the user modify any existing record.