Address Book DB From Schema

Let's build an address book so we can keep track of all the awesome programmers we know. Based on the **Address Book Schema** that you have designed just now, write a ruby program to create objects to represent the address book and create the database `address_book.db`

Objectives

Create the address_book database

Create the SQL Schema that represents your database tables. Then, create a file called `setup.rb` which you will run only once and will create your database. All other parts of your code will assume that the database has already been created.

Next, load some initial records into this database from hardcoded SQL statements. This code should also be in `setup.rb`.

Extra Credit : Instead of hardcoding your insert statements, create a CSV/YAML/JSON file with the data and import it.

Build the Ruby classes

First, create classes that correspond to your tables, e.g., a "contacts" table corresponds to a Contact class. These classes should have methods that allow you to add, delete, update records in the database.

It should work as follows:

```
contact = Contact.new(:name => "Alex")
contact.id # => nil
contact.save # => This executes an INSERT statement and makes a new record
contact.id # => 20


contact.name = "Bob"
contact.save # This executes an UPDATE statement
```

Note, you will also need to allow your classes to have access to your database. There are several ways to do this, but an easy way would be just to create a global variable that is accessible in all of your classes. (This is fine for now we'll get into more OO approaches later).

```
$db = SQLite3::Database.new "address_book.db"
```

Work with the address_book database using Ruby

Next, create driver code and additional methods in your code that allows you to complete the following tasks.

1. Add a contact
2. Add a group
3. Change a contact address
4. Delete a contact
5. Delete a group - careful here - what does this mean? What implications does it (should it) have on your data?

Data validation

Input from users can be "Dangerous". What if they give you a phone number with 17 digits? How about an address without a state? How does your database handle this? How does your Ruby code handle this?

These are complex questions. Come up with a plan to work with phone numbers (must have a valid format) and email addresses (must have a valid format and be unique).

Implement this plan in your code.