Report for Ruby Practical 9

Student Number: 16212138 Student Name: Qirun Chen Teacher Name: Mark Kean

Question 1

In this question, I think three tables should be created for storing information about users, books, and records of each borrowing. These three tables have associations with each other. This means that records are connected with users and books by user_id and book_id which are the primary key in user table and book table respectively.

I define an Active Record Migration class for creating records, and implement an Active Record Association to connect the record model with the user model and the book model. However, user_id and book_id in the record table should be defined as foreign keys which I do not implement in the code.

After creating some records, the data can be retrieved in the Database Management Client, like Navicat. The preview is in the picture below.

Book table

id	title	author
1	The Snow	Jo Nesbo
2	Kill Bill	Rogan
3	The Breal	Marian Ke

User table

id	name	age
1	Bernardo	21
2	Prof. Cynt	43
3	Mr. Donni	46

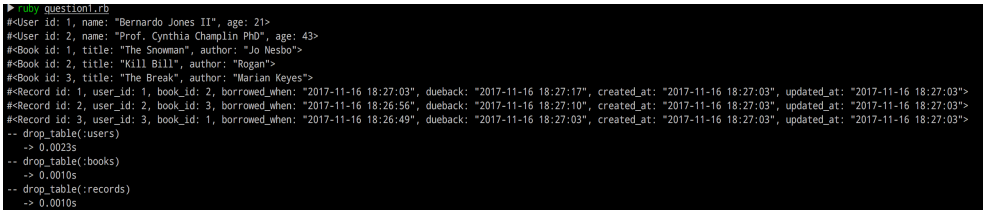
Record table

id	user_id	book_id	borrowed_when	dueback	created_at	updated_at
1	1	2	2017-11-16 18:27:03.200202	2017-11-16	2017-11-16 18	2017-11-16 18:
2	2	3	2017-11-16 18:26:56.202881	2017-11-16	2017-11-16 18	2017-11-16 18:
3	3	1	2017-11-16 18:26:49.204478	2017-11-16	2017-11-16 18	2017-11-16 18:

Furthermore, I use the "**find**" method to select records from tables, and print these records. However, the records printed by "p" function are empty. I suppose the reason of this is that the connection between SQLite and the Active Record entities is the initial cache.

```
▶ ruby question1.rb
-- create_table(:users)
   -> 0.0023s
-- create_table(:books)
   -> 0.0014s
-- create_table(:records)
   -> 0.0033s
#<User >
#<User >
#<Book >
#<Book >
#<Book >
#<Record >
#<Record >
#<Record >
```

If **the program is run again**, then it will load data from tables which already exist. In this case, the records printed by "p" function will be not empty.



When using the "find" method, I find it can retrieve records by record ids, and also **by an array with many record ids**, which is really powerful.

Question 2

I define a lambda object which prints the given parameter, and put this lambda with "&" in an "each block". Because of this, it is able to print records iteratively, or do some process on the given parameter. The code is presented in the picture below, and can be found in the folder with this report.

```ruby
users = User.find([1,2])
users.each(&print_records)
```

Besides, I optimize the way to create records by defining a lambda with multiple parameters. The reason why I do this is that the original version like in the picture below, where the parameters are assigned and passed by symbols, may lead to a mass order of assigning parameters like book2 and book1 in the code.

```ruby
book1 = Book.create(:title => "The Snowman", :author => "Jo Nesbo")
book2 = Book.create(:author => "Rogan", :title => "Kill Bill")
book3 = Book.create(:title => "The Break", :author => "Marian Keyes")
```

In contrast, the lambda object I implement below can **constraint on assigning parameters in consistent order**; I think the code I write is smart and elegant.

```ruby
user_create = -> name, age {User.create(:name => name, :age => age)}
user1 = user_create.("Bernardo Jones II", 21)
user2 = user_create.("Prof. Cynthia Champlin PhD", 43)
user3 = user_create.("Mr. Donnie O'Reilly", 46)
```