



Ruby on Rails Training - 2

SQL

Summary: SQL (Structured Query Language) is a normalized computer language used to manipulate relational databases. Rails provides excellent support for this system; it supports and relies on it. Therefore, it is important for you to understand its principles.

Version: 1

Contents

I	Preamble	2
II	General rules	3
III	Today's specific instructions	4
IV	Exercise 00: CRUD starts here	5
V	Exercise 01: Seeds and migrations	7
VI	Exercise 02: Create and read	9
VII	Exercise 03: Active Record Associations	11
VIII	Exercise 04: Dynamic creation	12
IX	Exercise 05 : Validation	13
X	Exercise 06: 3D	15
XI	Exercise 07: Model Methods	16
XII	Exercise 08: Select	17
XIII	Exercise 09: Scope	18
XIV	Exercise 10: CRUD End Here	19
XV	Submission and peer-evaluation	20

Chapter I

Preamble

01010111 01101000 01111001 00100000 00110100 00110010 00111111
00100010 01010100 01101000 01100101 00100000 01100001 01101110 01110011 01110111
01100101 01110010 00100000 01110100 01101111 00100000 01110100 01101000 01101001
01110011 00100000 01101001 01110011 00100000 01110110 01100101 01110010 01111001
00100000 01110011 01101001 01101101 01110000 01101100 01100101 00101110 00100000
01001001 01110100 00100000 01110111 01100001 01110011 00100000 01100001 00100000
01101010 01101111 01101011 01100101 00101110 00100000 01001001 01110100 00100000
01101000 01100001 01100100 00100000 01110100 01101111 00100000 01100010 01100101
00100000 01100001 00100000 01101110 01110101 01101101 01100010 01100101 01110010
00101100 00100000 01100001 01101110 00100000 01101111 01110010 01100100 01101001
01101110 01100001 01110010 01111001 00101100 00100000 01110011 01101101 01100001
01101100 01101100 01101001 01110011 01101000 00100000 01101110 01110101 01101101
01100010 01100101 01110010 00101100 00100000 01100001 01101110 01100100 00100000
01001001 00100000 01100011 01101000 01101111 01110011 01100101 00100000 01110100
01101000 01100001 01110100 00100000 01101111 01101110 01100101 00101110 00100000
01000010 01101001 01101110 01100001 01110010 01111001 00100000 01110010 01100101
01110000 01110010 01100101 01110011 01100101 01101110 01110100 01100001 01110100
01101001 01101111 01101110 01110011 00101100 00100000 01100010 01100001 01110011
01100101 00100000 00110001 00110011 00101100 00100000 01010100 01101001 01100010
01100101 01110100 01100001 01101110 00100000 01101101 01101111 01101110 01101011
01110011 00100000 01100001 01110010 01100101 00100000 01100001 01101100 01101100
00100000 01100011 01101111 01101101 01110000 01101100 01100101 01110100 01100101
00100000 01101110 01101111 01101110 01110011 01100101 01101110 01110011 01100101
00101110 00100000 01001001 00100000 01110011 01100001 01110100 00100000 01100001
01110100 00100000 01101101 01111001 00100000 01100100 01100101 01110011 01101011
00101100 00100000 01110011 01110100 01100001 01110010 01100101 01100100 00100000
01101001 01101110 01110100 01101111 00100000 01110100 01101000 01100101 00100000
01100111 01100001 01110010 01100100 01100101 01101110 00100000 01100001 01101110
01100100 00100000 01110100 01101000 01101111 01110101 01100111 01101000 01110100
00100000 00100111 00110100 00110010 00100000 01110111 01101001 01101100 01101100
00100000 01100100 01101111 00100111 00101110 00100000 01001001 00100000 01110100
01111001 01110000 01100101 01100100 00100000 01101001 01110100 00100000 01101111
01110101 01110100 00101110 00100000 01000101 01101110 01100100 00100000 01101111
01100110 00100000 01110011 01110100 01101111 01110010 01111001 00101110 00100010

Chapter II

General rules

- Your project must be realized in a virtual machine.
- Your virtual machine must have all the necessary software to complete your project. These softwares must be configured and installed.
- You can choose the operating system to use for your virtual machine.
- You must be able to use your virtual machine from a cluster computer.
- You must use a shared folder between your virtual machine and your host machine.
- During your evaluations you will use this folder to share with your repository.
- Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors. If this happens, your project will be considered non functional and will receive a 0 during the evaluation.
- We encourage you to create test programs for your project even though this work **won't have to be submitted and won't be graded**. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.
- Submit your work to your assigned git repository. Only the work in the git repository will be graded. If Deepthought is assigned to grade your work, it will be done after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.

Chapter III

Today's specific instructions

- All work for the day must be submitted in the provided Rails project.
- Any Gem not authorized by the instructions is prohibited.
- Any global variable not authorized by the instructions is prohibited.
- Tests are provided for all exercises, and you must pass the tests related to the current exercise.
- You must handle errors properly. No Rails error pages will be tolerated during the grading, even with incorrect values such as a nonexistent database or a poorly formatted table, etc.
- The tests and seed files should be left as they are.
- You should have NO offenses reported by Rubocop..
- You must LEAVE the “.rubocop.yml” file intact (it contains a customized Rubocop configuration specifically for d05).

Individual tests can be run for clarity using the following command:

```
ruby -I"lib:test" test/path\_to\_file.rb -n "file\_name"
```

Make sure to replace “name” with the name of the test to be executed. It's possible that the command may not work due to dependencies. This will not be accepted during grading, so you must know how to handle these errors.




NB: The sqlite3 lib is very sensitive to corruption. Feel free to reinstall it.



In today's module, you will come across a somewhat puzzling name, the "cuicui." It simply corresponds to the literal translation of "tweet" into French.

Chapter IV

Exercise 00: CRUD starts here

	Exercise 00
Exercise 00: CRUD starts here	
Turn-in directory : <i>ex00/</i>	
Files to turn in : Seek_well	
Allowed functions : \$db	

You will find some basic foundations and a controller with a few pre-named function. In order to formalize the submitted work, you must keep the names of the routes and methods.

Indeed, tests have been written for you. Moreover, you can run them using the following command:

```
rake test
```

You can also run one test at a time. We'll let you go through the doc. They've been named according to the methods they test, so you need to adapt.

Know that you're supposed to take all the tests within the day. Anyway, in order to score all the points during the evaluation, you will have to complete all the exercises... but will this be enough?

Alright, let's get started!.

Start the server of the provided application, go to `localhost:3000` and you will be granted with a wonderful list of buttons that... do nothing... for now.

For this exercise, we'll start gently and simply implement the code for the first three methods of the `ft_query` controller.

The first method: `'create_db'` should create the file that will be used by SQLite for the rest of the exercise. This file **MUST** be named `"ft_sql"`. The method should create an instance of the `SQLite3::Database` class and store it in the global variable `"$db"`. Yes, another global variable, but this time it's for the sake of scope (we think you've understood the purpose by now). You will have other global variables unlocked during the day. Also, no errors should appear if you click on `'create_db'` multiple times. The second method: `'create_table'` should create two tables in our `'ft_sql'` file. One is named `'clock_watch'`, and the other is named `'race'`. Both these tables include (in that order):

- `'clock_watch'` :
 - An auto-incremented primary key: `ts_id`
 - An integer: `day`
 - An integer: `month`
 - An integer: `year`
 - An integer: `hour`
 - An integer: `min`
 - An integer: `sec`
 - An integer: `race`
 - A string (50 Chars max): `name`
 - An integer: `lap`
- For `'race'`:
 - An auto-incremented primary key: `r_id`
 - A string (50 Chars max): `start`


The third method: `'drop_table'` should completely deletes the tables created by the `'create_table'` method.

Once again, no bug are be tolerated if you vigorously hammer the button.

To convince yourself that your methods work correctly, the command `rake test` should validate the tests `'db_file_creation'`, `'db_table_creation'` and `'db_drop_table'`. It's up to you to go and check the code!

Chapter V

Exercise 01: Seeds and migrations

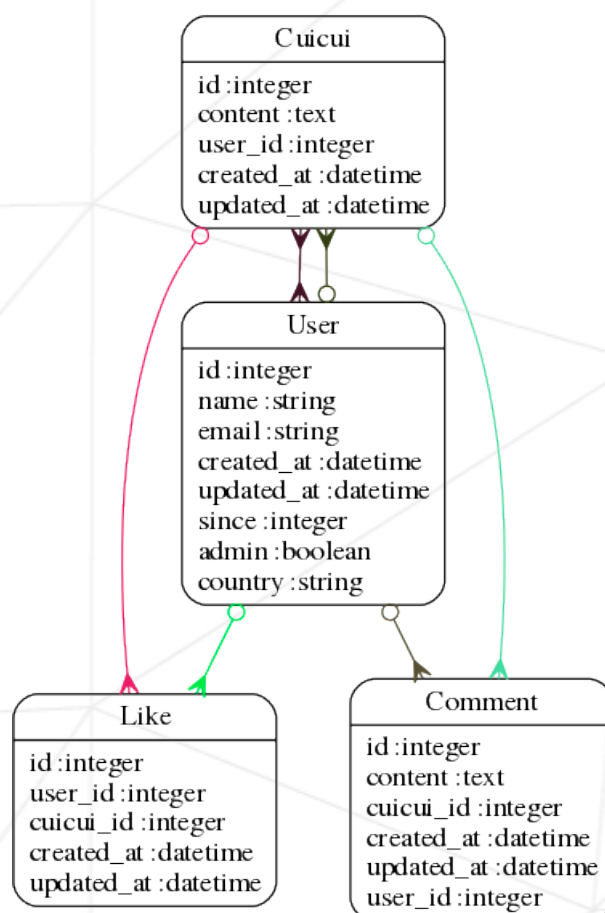
	Exercise 01
Exercise 01: Seeds and migrations	
Turn-in directory : <i>ex01/</i>	
Files to turn in : tweetos_aka_hello_rails	
Allowed functions :	

You are starting to grasp the concept of SQL, but in Rails, we benefit from a whole bunch of smart tools. For example, there are **migrations** which allow you to script the design of your tables in a versioned manner. Then there are **seeds** which allow you to script the initial population of your tables.

You can imagine how useful and important these two tools are. Here, you are provided with a growing application, and a seed is already present. Your task is to create the appropriate migrations that will create the tables required for the seed's population.

Also, specify a default route using the “root” macro in the “config/routes.rb” file. Be wise in your choice.

You need to create migrations specific to the database schema shown above:



For now, tables relationships are not required. However, to validate the exercise, you must pass the `migrations_test.rb` without any errors.


Use [GIT: Migrations](#) are sensitive and can be rolled back for modifications. But not always.



Learn about the different options of the 'rail generate' command. You will discover various generators similar to scaffold, that have the advantage of making you coffee and serving breakfast in bed. Now you know that... make your choice!

Chapter VI

Exercise 02: Create and read

	Exercise 02
Exercise 02: Create and read	
Turn-in directory : <i>ex02/</i>	
Files to turn in : Seek_well	
Allowed functions : <code>\$runner_1</code> , <code>\$runner_2</code> , <code>\$runner_3</code> , <code>\$runner_4</code> , <code>\$time_stamps</code>	

Are you ok? Did the migrations not overwhelm you? Perfect!.

Let's go back to pure and hardcore [SQL](#)...

Now let's focus on the "Subscribe / Start race" button. As you may have understood, clicking it once should have the following effects:

- in the "clock_watch" table
 - 4 new entries in clock_watch with the 4 names provided in the input boxes on the right.
 - The exact moment of the click for each entry, in the corresponding day, month, year, hour, min, sec fields.
 - The value of lap set to 0.
 - A unique id in the clock_watch table for ts_id.
 - A unique id for the race corresponding to the correct r_id.
- in the "race" table:
 - A unique id in its own table for r_id.
 - The value of start filled with the string from (Time.now) which should correspond to the value of the "0"lap created in clock_watch (the race starts when the runners begin).

You should assign the values from the four inputs to the 4 new global variables. If any names are missing, your method should automatically assign the fields with “anonymous”.

This means that by filling the first two input boxes with “foo” and “bar” and clicking “start” on a Sunday evening, you should have something like this:

```
table clock_watch:
  ts_id  day   month  year   hour  min  sec  race  name      lap
-----
3       3     7      2016   18    7    40   0     anonymous  0
2       3     7      2016   18    7    40   0     anonymous  0
1       3     7      2016   18    7    40   0     bar        0
0       3     7      2016   18    7    40   0     foo        0

table race :
r_id start
-----
0       2016-07-03 18:07:40 +0200
```

Anyway, the tests are there to tell you if you’re on the right track or not. When in doubt, remember that the test are always right.

If you use the variables correctly, you should see all 4 fields automatically fill in when you click “Subscribe / Start”.



NB: Retrieve the names passed as parameters by the button in your methods like this: `query_params[:name_1]`, `query_params[:name_2]` etc.

To view your results, uncomment the arrays in the view by removing the lines with the comment `<!-- uncomment at ex02 -->` :

```
in [your_path_to_app]/app/views/ft_query/index.html.erb:
.....
<%if false%> <!-- uncomment at ex02 -->
.....
<%end%> <!-- uncomment at ex02 -->
```


In the index method of the “ft_query” controller, fill the variable `$time_stamps` with the contents of the “clock_watch” table. If your table is empty or if the `ft_sql` file does not exist yet, fill the variable with an appropriate string, for example: `['Database is empty or another error occurred']`

For now, fill `$all` with `['Not so fast, young padawan']`.

You successfully complete the exercise if you pass the “start_race” test, follow all the instructions in the list above and your result looks like table below the list.

Chapter VII

Exercise 03: Active Record Associations

	Exercise 03
Exercise 03: Active Record Associations	
Turn-in directory : <i>ex03/</i>	
Files to turn in : tweetos_aka_hello_rails	
Allowed functions :	


Active Record Associations is not a random term.
Once again, to succeed in this exercise, you need to pass the following tests:

- Comment's relations methods
- Cuicui's relations methods
- Like's relations methods
- User's relations methods

To get an overview of the data used for the tests, take a look at “test/test_helper.rb”.
Indeed, the test database is independent from the development and production database.

Chapter VIII

Exercise 04: Dynamic creation

	Exercise 04
Exercise 04: Dynamic creation	
Turn-in directory : <i>ex04/</i>	
Files to turn in : Seek_well	
Allowed functions :	

Its delightful syntax as missed,so let's return to SQL.


You should implement the well-named method “insert_time_stamp”, called by the four “time” buttons, which adds an entry “clock_watch” with:

- A new unique ts_id
- A value for the “race” field that corresponds to the last id of the “race” table.
- The name that correspond to the box to the right of the “Time” button that has just been pressed.
- The value of lap, which is incremented. Each click on “Time” button, times the lap and counts the number of laps in the column titled “lap”.
- The exact moment of the click for each entry in the corresponding day, month, year, hour, min and sec fields.

To validate this exercise, your must fulfill the above conditions AND pass the “insert_time_stamp” test.

Chapter IX

Exercise 05 : Validation

	Exercise 05
Exercise 05 : Validation	
Turn-in directory : <i>ex05/</i>	
Files to turn in : tweetos_aka_hello_rails	
Allowed functions :	

All your models are now interconnected. You can now uncomment the views. There are many. To make your life easier, use the script at the root of the rails project and proceed as follows:

```
ruby views_com.rb unco
```

Contemplate all this fresh data that has never been used before, a golden opportunity! (If there are still errors, it's because the previous exercises that deal with this application are not correctly done.)

It has value because it is organized; some duplicates have already crept into the contents, and it could degenerate quickly with the provided forms. So, to protect your capital, you should create data validation. *#readTheRailsDocsNotComplicated*.

To succeed in you economic challenge, you must ensure:

- Uniqueness of users based on their name and email.
- Minimum name length of 2 characters.
- Prohibition of the following names: ("42", "lancelot du lac", "Ruby") with the message "<name> is banished".
- Presence of the name, email, since and country for each user.
- Uniqueness of likes in relation to the association IDs(only one like per cuicui and per user, which seems logical).

- Valid syntax for email.
- Presence of User_id on cuicui creation.
- Presence of User_id on comment creation.
- Presence of content on cuicui creation.
- Presence of content on comment creation.
- Uniqueness of content for cuicuis.
- Uniqueness of contents for comments.
- All IDs must be strictly numeric.
- All IDs must be present.
- All IDs must be unique.
- The IDs must be valid (existing for the associated model).

As the IDs of associations are manually entered, for example, "http://localhost:3000/cuicuis/new" allows you to enter an invalid string or ID in the "User" box. Therefore, you also need to ensure the validity of this information. So, whether it's for updating or creating new records, and for likes, comments, and cuicuis, you must create validations that ensure the associated IDs exist.


In other words, you need to implement validations in your application's code that check whether the provided IDs for associations (likes, comments, cuicuis, etc.) actually correspond to existing records in the respective associated models (e.g., users). This will help maintain data integrity and prevent errors when creating or updating records with incorrect association IDs.

Your information bank will be well guarded and of guaranteed quality over time, eliminating the need for constant monitoring/moderation.

The tests for this exercise can be found in "test/models/*".

Chapter X

Exercise 06: 3D

	Exercise 06
Exercise 06: 3D	
Turn-in directory : <i>ex06/</i>	
Files to turn in : Seek_well	
Allowed functions :	


You have already implemented '**drop_table**' and now you are going to implement the code for the '**delete_all**' and '**delete_last**' methods.

- “delete_last” destroys the last record in the “time_stamp” table.
- “delete_all” destroys ALL the rows in the table.

Tests are also present for this exercise.

Chapter XI

Exercise 07: Model Methods

	Exercise 07
Exercise 07: Model Methods	
Turn-in directory : <i>ex07/</i>	
Files to turn in : tweetos_aka_hello_rails	
Allowed functions :	

Now you can uncomment the last section “`http://localhost:3000/users/<userid>`”, which corresponds to the “show” page of the user whose ID is placed instead of `<userid>` in the above URL.


Remove the lines from the corresponding views where it is written:
<!-- uncomment in ex07 -->

You have many errors, and that’s normal because your exercise is not finished. You should open the User model and implement these methods:

- `fame`: Returns an integer that is the sum of all the likes given to all the `cuicuis` of the user.
- `senior?`: Returns true if the “since” is more than 10 years old.
- `junior?`: Returns true if the “since” is less than 10 years old.
- `responses`: Lists the 5 latest comments on the user’s `cuicuis` (the last in creation, not in modification).
- `top_cuicui`: Lists the user’s `cuicuis` sorted by the number of likes.

Chapter XII

Exercise 08: Select

	Exercise 06
Exercise 08: Select	
Turn-in directory : <i>ex06/</i>	
Files to turn in : Seek_well	
Allowed functions : \$all	

You are getting used to it, and now you are implementing the code for the “all_by_name” and “all_by_race” methods. Additionally, a new global variable “\$all” is provided for you.


You must ensure that:

- all_by_name: Stores all entries from your “time_stamp” table sorted by runner’s names in the global variable “\$all”.
- all_by_race: Stores all entries of your “time_stamp” table sorted by the race ID in the global variable “\$all”.

There are tests available to help you. It’s up to you to play and have no errors.

Chapter XIII

Exercise 09: Scope


	Exercise 09
Exercise 09: Scope	
Turn-in directory : <i>ex09/</i>	
Files to turn in : tweetos_aka_hello_rails	
Allowed functions :	

In the cuicui model, at the line: "**# scope :top, lambda implement here**", replace “implement here” with your code so that on the page the “like” button leads to, the list of the most liked cuicuis is displayed.

Use the tests to understand the composition of the expected collection.

Chapter XIV

Exercise 10: CRUD End Here

	Exercise 10
Exercise 10: CRUD End Here	
Turn-in directory : <i>ex10/</i>	
Files to turn in : Seek_well	
Allowed functions :	

You are going to implement the last step of the CRUD: Update

Your runners should be able to change their names on the fly for all the `time_stamps` of the current race, except if they were anonymous. In that case, their name will remain unchanged.

Tests are available to help you with this exercise.

Chapter XV

Submission and peer-evaluation

Submit your work to your `Git` repository as usual. Only the work present in your repository will be evaluated during the defense. Make sure to check the names of your folders and files to ensure they comply with the subject's requirements.



The evaluation process will take place on the computer of the evaluated group.