Lab 5: Uploading the Data

Due date: Wednesday, February 15, in-class.

This is a **team lab**. Each group submits one set of deliverables.

Lab Overview

While most of your project development will happen outside the lab, we use this lab assignment to start the software development. You will develop the code to address the use cases that load the data into the database, in particular, the bulk-loading use cases.

Data

In support of this lab, and your futher software development activities, we are releasing the test dataset that you will use throughout the course. The dataset consists of a number of .CSV (comma-separated values) files representing three types of data:

- Job Parameters CSV files. These files contain the information about the TESS queries made to discover the regulatory elements associated with a single gene in a single experiment. The information includes the data about the experiment itself[1], as well as about the specific parameter settings that were used to ask TESS for data.
- Sequences CSV files. These files contains the promoter region sequence data for the genes.
- Hits1 CSV files. These files contain information about regulatory elements found in the promoter regions of a given gene in a given experiment.

[1] This particular information is repeated across multiple Job Parameters files for the same experiment.

File Naming Conventions. For each experiment and each gene discovered to be different in abundance in the experiment, there are three CSV files provided to you. Each file starts with the name of the gene, followed by the term `"promoter TESS"` followed by the type of the file (`"Job Parameters"`, `"Sequences"`, `"Hits 1"`). For example, the information about the `bASS1` gene for the experiment, will be found in the following files:

```
bASS1 promoter TESS Job Parameters.csv
bASS1 promoter TESS Sequences.csv
bASS1 promoter TESS Hits 1.csv
```

(note the spaces).

Assignment

Each team shall implement complete preliminary versions of the following use cases from the current version of the Use Cases document:

1. **UC-1.**
2. **UC-2.**

"Complete preliminary" version means the following:

- **(Complete)** The full functionality of the use case shall be implemented.
- **(Preliminary)** The design of the actual web page/pages or GUI(s) implementing each use case does not have to be complete/final.

Implementation notes

What is expected. Use case implementations shall have UI associated with them. Given the nature of the project, I expect that each task specified above can be accomplished either

- by typing a URL on the web server of a team, loading the web form at that URL, providing the necessary information (file/directory names) and hitting the "Upload" button or its analog; or
- by starting a program, which produces a dialog window implementing the upload functionality.

Clear All button. To make testing and retesting easy, put on the front page of your web application/front window of your desktop application a "Clear Data" button. When this button is pressed, your program shall execute a series of `DELETE FROM <Table>;` commands for each table in your database[2] to clean up the contents of the database. (You will, of course, remove this functionality later.)

CREATE TABLE statements. Your software shall work under the assumption that the database has been created. That is, you will run your `CREATE TABLE` statements off-line to create the database, before deploying the web-based software.

Error handling. Obvious errors (missing files/directories) shall be recognized and dealt with graciously. You can assume no errors in the CSV data, but if the formats are wrong (e.g., someone mistakingly put freezer log file name, where the pyroprint log file name was supposed to be), you need to be able to tell that and go back to the "waiting for input" mode.

While this is not really "error handling" per se, your code, from the very beginning shall have at least basic protection against SQL injection and buffer overrun attacks.

Parsing data in the HITS1 data files. There are two columns in the HITS 1 data files that contain complex data. Use the following instructions when parsing the data.

Factor. This column can have one of two formats.

1. Whenever the value of the `Model` column is an "I-number" (i.e., a value that starts with an `I`), the format of the `Factor` column is straightforward. The column will contain two values in it. The first value is **always** going to be `"_00000"` for this case, and it represents lack of information about the so called "T-numbers" (see below). This value can be ignored. The second value of in the column, separated from the first by a space is a string value representing the actual name of the transcription factor that binds to the regulatory element described by the current row.

   Some examples of the contents of the `Factor` column that work this way:

   ```
   _00000 c-Jun
   _00000 TEF1/GT-IIC
   _00000 C/EBP
   _00000 RXR-alpha
   ```

[2] If your database contains any tables that do not participate in bulk insert procedures, feel free to leave them off the list.

These describe transcription factors with names `c-Jun`, `TEF1/GT-IIC`, `C/EBP` and `RXT-alpha` (note the inclusion of dashes, slashes and possibly other punctuation in the name of the transcription factor).

2. Whenever the value of the `Model` column is an "M-number" or one or more "R-numbers" (see more on "R-numbers" below), i.e., the name of the model (models) starts with a letter "M" or a letter "R", the `Factor` column will have the following format:

   The column shall contain multiple **space-separated** values of two types:

   (a) T-numbers. A "T-number" is a reference to a resource page maintained by a TESS database. T-numbers have the following format: they start with a letter `"T"` followed by **exactly five digits**.

   (b) Transcription factor names. Any identifier/value on the list that does not parse as a valid "T-number" shall be considered a name of a transcription factor.

   The first value in such cells will always be a T-number. At least one T-number and at least one transcription factor name will be present in each cell.

   Examples of the contents of the `Factor` column of this type include:

```
T00311 GATA-3
T00138 T00139 T00137 c-Myb
T00032 c-Fos T00122 T00123 T00124 T00125 T00131 T00132 T00133 T00134 T01156 c-Jun
T01113 ELF-1 T01418 NF-CLE0a T01419 NF-CLE0b
```

   The first example shows one T-number (`T00311`) and one transcription factor (`GATA-3`); the second example represents one transcription factor, `c-Myb` and multiple (three) T-numbers. The third example has 10 T-numbers and two transcription factors: `c-Fos` and `c-Jun` while the last examples shows three T-numbers and three transcription factor names (`ELF-1, NF-CLE0a` and `NF-CLE0b`).

   Each value needs to be parsed and stored as a separate association with the current regulatory element.

Model column. Model column can have three different formats, but we only separate them into two different formats.

1. **I-numbers and M-numbers.** Whenever a model name starts with an `"I"` or an `"M"`, the format of the `Model` column is going to be as follows.

   The column will include two values. The first one is the model number (an I-number or an M-number). The second value is a reference to the transcription factor name *in parentheses*. The transcription factor name reference will be in different format for I-numbers (same string as the transcription factor name from the `Factor` column) and M-numbers (a value that starts with a `"V$"`). **But, for parsing purposes, you can ignore this value.** Transcription factor information shall be established based on the contents of the `Factor` column.

Examples of the values of this type are:

```
I00216 (TFII-I)
I00239 (Elf-1/NTF-1)
M00077 (V$GATA3_01)
M00160 (V$SRY_02)
```

2. **R-numbers.** R-numbers are model names that start with a letter `"R"`. A single `Model` cell in the CSV file can contain multiple R-values in it. The format of such a cell is going to be as follows:

   - An R-number,
   - followed by `()`,
   - optionally followed by one or more repetitions of the R-number, `"()"` pattern.

   Examples of R-number occurrences in the `Model` column are:

```
R03584 ()
R00700 () R00701 () R00703 () R00808 () R01349 () R02144 () R02921 ()
```

   These values shall be represented as a single value of the `Model` field in your database table. Essentially, you are asked to parse individual R-numbers and concatenate them together using a `"/"` delimiter to separate two R-numbers from each other.

   For the examples above, your parser shall convert the values of the `Model` column into the following strings:

```
'R03584
'R00700/R00701/R00703/R00808/R01349/R02144/R02921'
```

   Please note, that we do not know up front how many R-values there are going to be. Use caution and observations (look into the CSV files) to determine the necessary number of bytes you need to be able to store all the data.

Deliverables

You will have some hard-copy deliverables and the software deliverable.

Code. Your main deliverable is operational code implementing the specified use cases. For web applications, ut a link to the front page of your application on the team wiki page. For desktop applications, upload the code to the code repository, put a link and a short description (README-file-style) on the team's wiki page.

Old hardcopy deliverables. Submit the entire trail of your project documentation from Labs 2,3 and 4. This will yield a numeric grade for your Lab 4 performance.

New hardcopy deliverable. Submit the final version of your database design. A `.sql` file of your `CREATE TABLE` statements will suffice.

ChangeLog. Submit a changelog documenting any changes in the database design from the point of your last set of deliverables. If no changes were made, submit a change log specifying that.

On the wiki. To finish off the design stage, post the final version of your E-R diagram and the `CREATE TABLE` statements on the team wiki page.

Demo. Each team will demonstrate the operation of the data loading functions during the Wednesday, February 15 lab period. Be ready to show (a) an empty database, (b) the operation of your web/desktop application, (c) the filled database after information was added to it.