Online Publication Management System

CMPE 138 Individual Project

Objective: Implement a simple web service that manages real scientific publications in Computer Engineering and allows users to query the database, add, and delete records from it.

Team size: 1 (Individual project – no collaborations with your friends)

General description: This project is relatively simple but gives you a chance to practice the essentials you learn from this course. You are given an XML data file that contains publications extracted from a previous system. Design a database that should hold these publications and a web service around it to allow querying the database. The following tasks should be completed:

1. Analyze the pubs.txt file and decide an appropriate database schema. Create an ER diagram for the database, which you will include in the report.
2. Create the database and its tables. The database should be an SQLite database. You should be able to answer the requisite queries correctly, and hopefully efficiently.
3. Write a script that reads the pubs.txt file, parses it, and insets the data in your database tables. The pubs.txt file contains many publications. Each <pub> ... </pub> pair specifies a publication, with ID, title, year, journal (<booktitle>), pages, and authors information. Some information may be missing. It is your own choice to use a default value or NULL for missing fields. Some information looks incorrect but you do not have to worry about it. The data in pubs.txt was automatically extracted from web resources by a script.
4. Create a Jupiter Notebook that creates the database, uses your script to populate the database, and executes the following queries:
   1) Insert a new publication into the database.
   2) Query all publications by a particular author.
   3) Query all publications in a particular year.
   4) Query all publications published in a particular journal.
   5) Query all publications with a particular string contained in the title.
   6) Query all publications with a combination of the author, year, journal, and title constraints.
   7) Delete a publication for a particular author, year, journal.
   8) Delete all records for a given author.
   9) Change the name of an author or journal.
   10) Change the title of an article or its publication year.
5. Create a web service that allows a consumer (generally a Web app) to ask the same kinds of questions as in the previous task. Users should be allowed to specify the format the publications should be returned in (e.g., XML, json), and how the publications should be sorted. In particular, the publications can be sorted by year, or journal, or author, or title, in either increasing or decreasing order. The total number of returned publications should be included in the payload as well. Finally, the user should be able to specify they only want a subset of the answer set (e.g., return items 31-60 from the list of results), which would allow implementing pagination of results in the web app.
6. Write a report in no more than 4 pages, in which you describe the design of the database (including ER diagram), functionality and features, and other aspects of the architecture or implementation that you think are worth mentioning.

7. (**Extra credit**) Implement a web app that consumes the web service you created and takes advantage of all the functionality the web service has to offer.

Deliverables: The following items should be included in a *zip* or *tar.gz* archive and uploaded to Canvas. The archive should be called *project.zip* or *project.tar.gz*.

- A python 2.7 script, named *create.py*, which can be called from the command line to create your database and load it with the publication samples in pubs.txt. The script should accept one argument (the publication file path) and should create a single database file, called *database.db*. Example execution:
  ```
  python create.py pubs.txt
  ```
- Your Jupiter notebook file, named *notebook.ipynb*, and a *pdf* version of your notebook, named *notebook.pdf*, showing the output of task 4 above. You can download the notebook as *html* and then print it to PDF (Chrome and other browsers have this built-in). In the notebook, you can assume the database has already been created and the file *database.db* exists in the same directory as the notebook.
- A *pdf* version of your report, named *report.pdf*.
- A sub-directory, named *service*, containing the code for the web service in task 5 above. You should include a README.txt file with instructions on running the web service.
- Optionally, a sub-directory named *web-app*, containing the code for the web app in the extra credit task above. You should include a README.txt file with instructions on running/installing the web app.

Evaluation: The projects will be evaluated based on database design, correctness, documentation, and appropriate evaluation/testing. The following grading rubric will be used for the project.

- Database design (40 pts)
  - Does it capture all the attributes correctly?
  - Does it allow efficient queries without unnecessary data duplication?
  - Is the ER diagram correctly drawn? Does it capture necessary constraints?
  - Is the diagram correctly translated into SQLite database tables? Does the *create.py* script work as expected?
  - Does the schema support the required queries?
- Database queries (20 pts)
  - Do all the requisite queries return appropriate data?
- Reporting (20 pts)
  - Does the report file exist and include all appropriate information?
  - Does the notebook file exist and execute properly? Is the pdf version of the notebook complete? Does it show the same output as the execution of the notebook?
- Service (10 pts)
  - Does the service include all the required functionality?
  - Are instructions for its execution complete? Does the service work?
- Package (10 pts)
  - Are all the files/directories in the archive named correctly?
  - Are there missing files?
- Web app (extra credit, 10 pts)