CSE-5331 | PROJECT 2

Description

In this project, we learn how to export data from a Flat/Relational to a Document-oriented format (JSON and XML), and import this to MongoDB. The objective of this project is to understand the differences of storing data in a RDBMS vs Document-based NOSQL System. Additionally, we look at parsing the JSON from MongoDB into an XML which is often useful in while building APIs and even migrating to another RDBMS.

Contributors

Student ID	Student Name	Contribution
1001767678	Harshavardhan Ramamurthy	MongoDB setup, create-table script, clear_data(), load_mysql_table(), Project-as-root, load_to_mongodb(), fetch_as_relational(), Department as root
1001767677	Karan Rajpal	MySQL setup, Documentation, format_as_xml(), Main-driver, Employee-as-root

Environment

1. Python - 3.7
2. Libraries
   - pymysql~=0.9.3
   - dnspython~=1.16.0
   - pymongo~=3.9.0
   - tabulate~=0.8.3
3. MySQL - 8
4. MongoDB - 4.2.8

Project Workflow

1. Data from text-files are loaded to their respective tables in MySQL
2. Data is retrieved from MySQL using `JOIN`s
3. Data retrieved in Document format(JSON) with `PROJECT` and `EMPLOYEE` respectively as root
4. Data in document format is loaded to MongoDB and retrieved

Project structure

The data required by the `DEPARTMENT`, `DEPT_LOCATIONS`, `EMPLOYEE`, `PROJECT`, and `WORKS_ON` tables are present in the respective text files under the `data/` directory

The SQL scripts required to create the necessary tables, query results in JSON format with nested objects are in the `scripts/` directory.

`PROJECT` as root

In relational format, the project name, number & department name are redundant for every employee that works on it.
Therefore the employee details are nested in the `project` document using `JSON_ARRAYAGG` function in
SQL by `GROUP`ing `BY` project name, number & department name.

`EMPLOYEE` as root

In relational format, the employee lname, fname, department name are redundant for every project that the employee works on.
Therefore the project details are nested in the `employee` document using `JSON_ARRAYAGG` function in
SQL by `GROUP`ing `BY` employee lname, fname, department name.

Formatting result as XML

Data retrieved in `JSON` format contains nested JSON objects for Employee and Project details respectively for
Project and Employee as root. These result-sets are then iterated over to form an XML document that conforms to the same
nested structure as the `JSON` result-set in the `format_as_xml()`

`DEPARTMENT` as root

Since we need to retrieve details of

1. `EMPLOYEE` who manages the `DEPARTMENT` and
2. `EMPLOYEE`s who belong to that `DEPARTMENT`

The `EMPLOYEE` table is `JOIN`ed twice - once based on `mgr_ssn` and next based on `Essn` and `JSON_ARRAYAGG` is used
in the same query to nest details of the `EMPLOYEE`s who belong to each department by `GROUP`ing `BY` dept number, name,
mgr_lname and mgr_fname

Querying from MongoDB

Documents with `PROJECT`,`EMPLOPYEE` and `DEPARTMENT` as root with their respecitve nested objects are queried from MongoDB using `collection.findall()` method.

Instructions

> IMPORTANT NOTICE: This project relies heavily on file-names. Please DO NOT change any filename in the project.
>
> For demo purpose, we've set-up MySQL and MongoDB on cloud with the necessary tables created and sufficient user-privileges so you could just do steps 1 and 2 to run the project

1. Install the necessary dependencies by executing `pip install -r requirements.txt` in the console

2. Execute the program by running `python main.py` in the console and follow the instructions on the screen

If you want to test/run the project with your own/different instance of MySQL and MongoDB, then

1. Create a dedicated database in MySQL and MongoDB for this project
2. Create the `DEPARTMENT`, `DEPT_LOCATIONS`, `EMPLOYEE`, `PROJECT`, and `WORKS_ON` tables using `scripts/create-tables.sql`
3. Ensure that the `user` has sufficient privileges(`INSERT` and `DELETE`) on the database in both MySQL and MongoDB.
4. Add the credentials for MySQL and MongoDB in `config.py`.
5. Execute the program by running `python main.py` in the console and follow the instructions on the screen

References

1. `JSON_ARRAYAGG`
2. `JSON_OBJECT`

Output

> Once the project is run, it prompts the user to press ENTER at various stages. You can verify that the data is loaded to the said data-store by querying it before pressing the ENTER key

A sample output is saved in `output.txt` file as it is quite long.