CS 3200 Database Design Final Project: Creature Spotting DBMS
Group Name: TabalonJBlancaflorFHsuT
Names: Jaana Tabalon, Florenz Blancaflor, Taylor Hsu

README section

See repo, README.md file, or README section below.

Summary

Our project was built to allow users to record, modify, and read occurrence records of certain flora and fauna found in the world. This project is meant to act as a management and analysis interface with which users can use to record and search specific creatures and track their appearances in certain areas.

We scraped the Global Biodiversity Information Facility (GBIF) web browser, which contains free and open access biodiversity data, to find real-life data about creatures, namely frogs, fish, trees, mushrooms, and flowers, found throughout the Philippines. These data records were used to populate the tables in our database.

In Our Repo

- */creature data:* Folder containing all of the CSV files scraped from the GBIF web browser used to populate our tables
- */sql:* Folder containing all of the SQL files created for this project including:
- *creatureSpottingsDB.sql:* SQL file to create all normalized tables in the database
- *creatureSpottingsDump.sql:* SQL file to create the entire database, including all table data, all at once
- *procs_triggers.sql:* SQL file to create all necessary procedures and triggers
- *CS3200creature_spottings.py:* The executable Python program that connects to the creature_spottings database and used to interact with the database
- *sqlVar.py:* Python file to hold variables of SQL calls made in our code
- *README.md:* Markdown file to explain our repo

Technology Needed

- MySQL Workbench (Installation link)
- Command-line interface
- Python (Installation link)
- PyMySQL (Installation link)
- Tabulate (Installation link)

Using Our Repo

Download the repo

1. Clone this repo onto your personal machine or download a ZIP file. If needed, unzip the ZIP file.
2. Save the repo to an easily findable location, like /Desktop, /Documents, or some other folder.

Load the DB dump

1. Open the file creatureSpottingsDump.sql in MySQL Workbench. (Download MySQL Workbench here.)
2. Run the entirety of the file to ensure you have all of the necessary tables, procedures, triggers, functions, and data to interact with this project.

Interacting

1. Open your machine's command-line interface (Terminal on Mac, Command Line on Windows, etc.).
2. Navigate to where you saved the copy of this repo. (i.e. cd Documents/CS3200-FinalProject)
3. To run this application, you will need the following installed on your machine: Python (at least v3.7), PyMySql, and Tabulate.
   Download Python here.
   Download PyMySql by running pip install PyMySQL in your command line interface.
   Download Tabulate by running pip install tabulate in your command line interface.
4. To begin using this application, type python CS3200creature_spottings.py into the command line interface and begin interacting by following the given prompts. If successful, you should see a prompt requesting you to enter your DB credentials.

Technical Specifications

We utilized SQL queries and Python as the host language in which the SQL queries were embedded into. We connected the SQL database to Python using the PyMySQL client library.

Below are the steps to access our repo and database and how to interact with our application.

Download the repo

3. Clone this repo onto your personal machine or download a ZIP file. If needed, unzip the ZIP file.
4. Save the repo to an easily findable location, like /Desktop, /Documents, or some other folder.

Load the DB dump

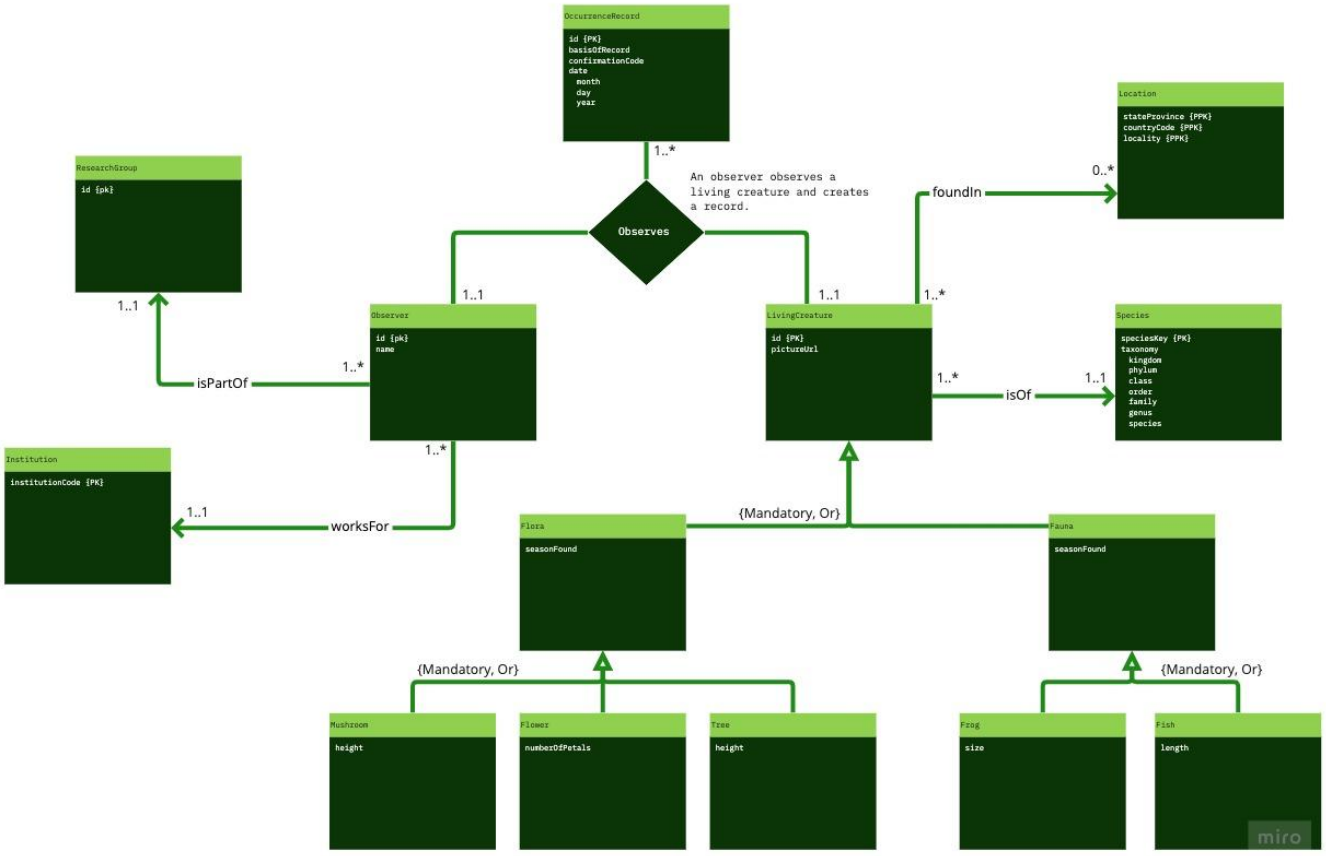
3. Open the file creatureSpottingsDump.sql in MySQL Workbench. (Download MySQL Workbench here.)
4. Run the entirety of the file to ensure you have all of the necessary tables, procedures, triggers, functions, and data to interact with this project.

Interacting

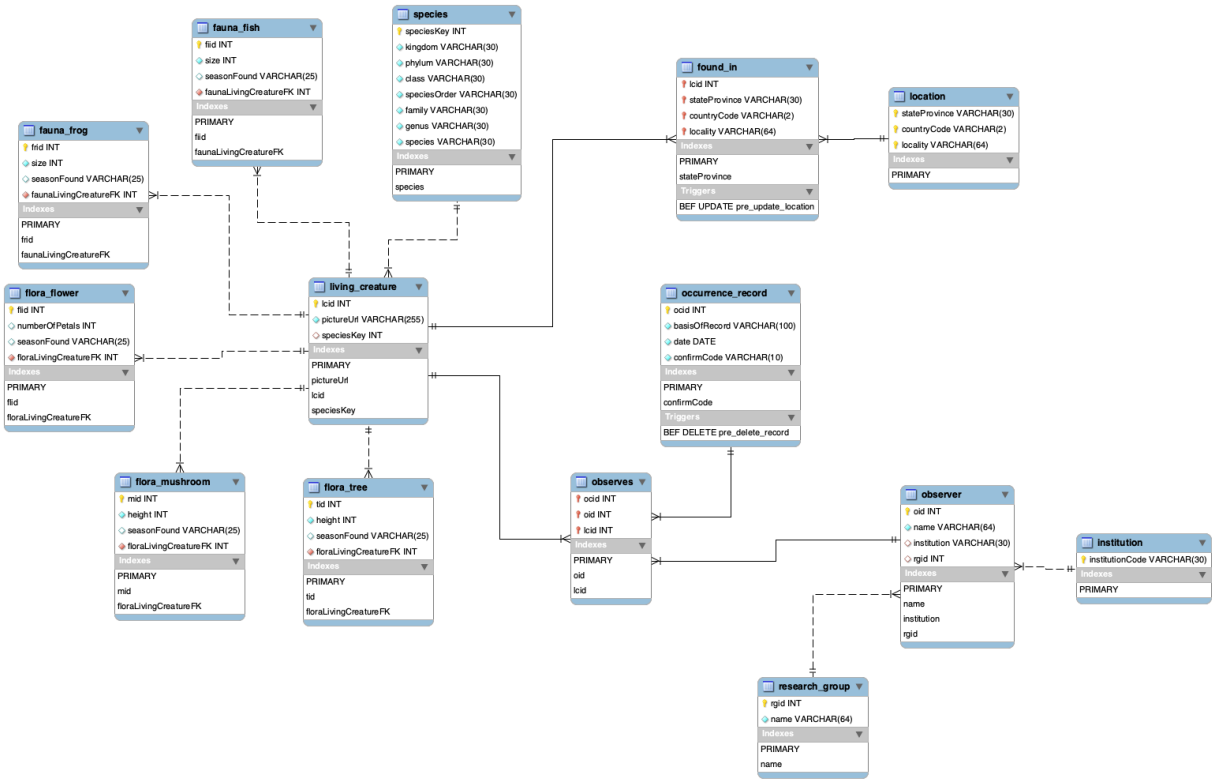
5. Open your machine's command-line interface (Terminal on Mac, Command Line on Windows, etc.).
6. Navigate to where you saved the copy of this repo. (i.e. cd Documents/CS3200-FinalProject)
7. To run this application, you will need the following installed on your machine: Python (at least v3.7), PyMySql, and Tabulate.
   Download Python here.
   Download PyMySql by running pip install PyMySQL in your command line interface.
   Download Tabulate by running pip install tabulate in your command line interface.

8. To begin using this application, type python CS3200creature_spottings.py into the command line interface and begin interacting by following the given prompts. If successful, you should see a prompt requesting you to enter your DB credentials.

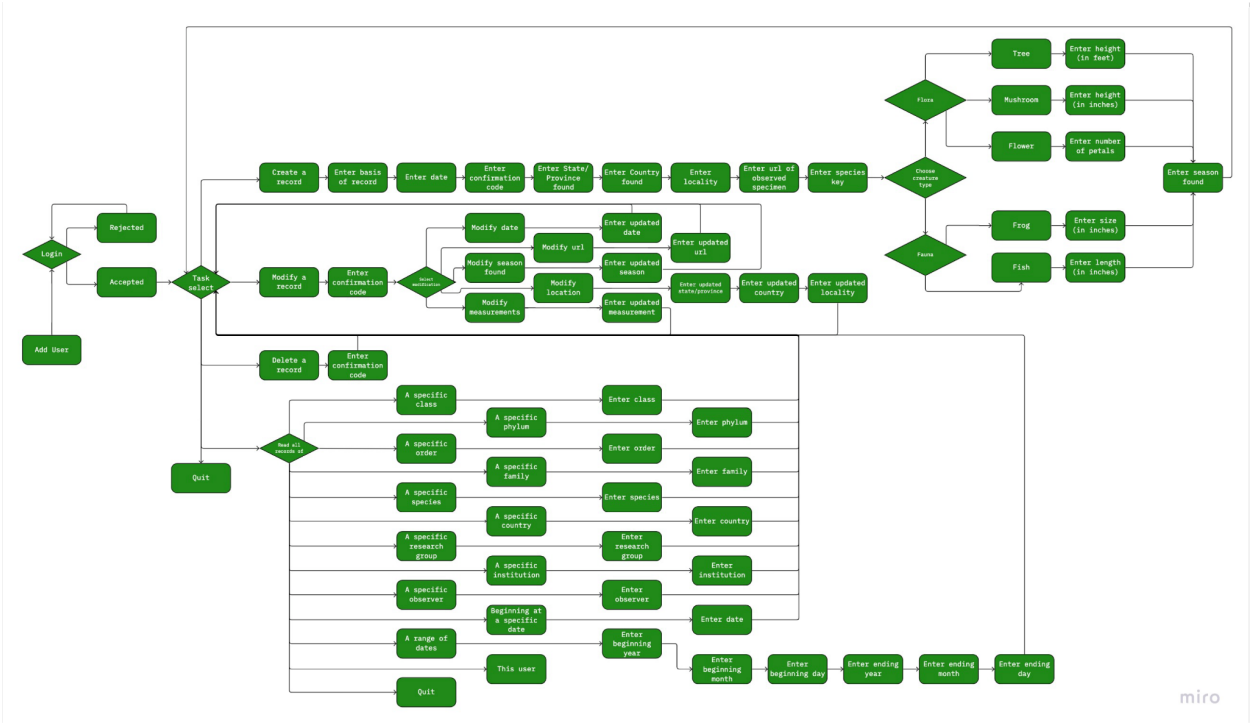
Conceptual UML Design



Logical Design



Final User Flow



Lessons Learned:

Technical expertise gained

The primary hurdle for us to overcome was learning how to create connections between the database, database procedures and triggers, as well as the Python file that defines how the user can work with the database through the command line interface. The members of our group each had different levels of experience with different aspects of programming. For example, some members had more experience working with Java and command line interfaces, whilst others had more experience using Python. For all three of us though, we gained technical expertise in programming in Python in a database context. Although we all had previous coding experience, this project was our first time utilizing code to create, manage, and read from a relational database.

We also gained technical expertise in writing complex SQL queries and practicing all of the SQL concepts we learned this semester. Due to the complexity of our database schema, we had to write many long and complicated SQL queries in order to create, delete, modify, and read tuples. This project gave us the opportunity to practice combining all of the skills we have learned this semester as well as learning how to troubleshoot and problem solve through errors we encountered.

Insights, time management insights, data domain insights etc.

As we continued to work on the project, the intricacies and difficulties of creating a database as detailed as this became more apparent. As we worked on each individual aspect, we found that more procedures, triggers, and code blocks needed to be added in order to ensure that the correct tuples were being pulled, added, and deleted from the database. Additionally as a result, we found that time management became more difficult. The more back-end functions, triggers, and procedures that were added to the database, the more it became difficult to account for how much additional time needed to be added to complete the project. Our group ended up finding it more effective to problem-solve together by working through the same functions at the same time. Working collaboratively allowed us to spend less time confused individually and reduced the time it took to get a function working properly.

The data domain of our project involved biodiversity data. By working through this project, our group realized that there is a lot of complex data related to recording and reporting wildlife, including photos, taxonomy classifications, quantitative measurements, spatial and temporal data, and much more. Although our database only contained 5 categories of wildlife, we realized how much more complicated a database can get with every additional category of wildlife or with every additional table. Also, with scientific data input by humans, we realized that there should be a high level of verification and security to ensure that users are not inputting invalid or incorrect information. When your database relates to scientific knowledge and research, ensuring that your database remains consistent and valid is key.

Related to verification and security, we focused heavily on database security and providing user indicators of exactly what could be done by the logged-in user in the database. As the intention of the database is to function as a research database we felt it was important to restrict who could perform different functions of the database. As a result, we learned that we should implement some function to check a user's identity and credentials. Users are allowed to create a new user profile and log in, but only verified users are allowed to perform operations aside from reading the database.

Realized or contemplated alternative design / approaches to the project

One design approach that we considered adjusting was the organization of our tables. As our schema stands right now, the occurrence_record and observes tables are separate. The occurrence_record table is broken down by different attributes, the occurrence record ID, the basis of record, the date, and the confirmation code. The observes table on the other hand contains the occurrence ID, the observer ID, as well as the living creature ID. Separating the table in this manner resulted in some back-end confusion when attempting to add triggers as these tables are associated through foreign keys, and as a result the tuples reference each other, making it difficult to perform delete operations. We thought that making this change would help simplify our schema as well as some of our queries, but in the end, however, due to time constraints, we were not able to make this design change.

Another potential way in which we could have approached our problem differently might be to utilize a NoSQL database instead. An example of a NoSQL database we might have used is a graph database, which allows users to store, map, and query data using nodes and creating connections by drawing edges between the nodes. Another option of a NoSQL database might be an object-oriented database, which means the data, as well as all of its attributes are tied together as an object. Implementing a NoSQL database would have allowed us to hold more data and access all of the data at a faster rate.

Another way we might have adjusted the design of the project might be developing the front-end user interface a little more. Rather than just utilizing a command-line interface there is potential to create a more visually interactive interface through creating a GUI interface system, like a web interface or desktop application interface. Having a more complex front-end interface would likely be more approachable to users. Especially considering the fact that, due to the nature of our database, our users would likely not have a very strong technical background, having a visual GUI interface through a website or application would make our project more approachable and usable.

Document any code not working in this section

As we began to incorporate more back-end triggers and procedures into our code we ran into a logical error while getting the delete trigger to work. When developing our initial schema, we felt it made sense to separate our observes and occurrence_record table. The occurrence_record table records data relating to the record, while the observes table matches the record ID to the specific observer and living creature instance. The goal of this trigger was to delete the record of the creature from the occurrence_record, observes, living_creature, and the table specific to what type of flora or fauna the creature is. However, as a result of the way we set up the schema, the trigger is being run before the deletion of the occurrence record. The observes table has a foreign key to the occurrence_record table, which means that we will be unable to delete the observes tuple within the trigger as it is referenced by the tuple the DELETE operation is being conducted on. In order to address this, it is likely that the observes and occurrence_record tables need to be merged together, which is a design consideration we unfortunately did not have time to implement.

Future Work:

Planned uses of the database

The intention of the database as we approached it initially was to have it function as research record storage. That means allowing a user to look up information based on a particular taxonomic classification, dates recorded, institutions, research groups, or users. Users can read observation records of the 5 sub-classifications of flora and fauna that we defined. Those 5 sub-classifications being mushroom, tree, flower, fish, and frog. On top of the read operations, this database allows for the user to complete any of the other 3 CRUD operations. Users can create a new user as well as a new occurrence record of some living creature. They can update the date, picture URL, season a creature was found, location, and creature measurements of an observation record. Finally, users can delete an existing occurrence record.

In terms of the future, there is potential for this database to be developed further by incorporating prediction queries. For example, allowing the user to generate a score that indicates whether a species of frog may be more likely to show up in a certain area or other useful metrics for research. Incorporating SQL prediction allows the researcher to perform queries that incorporate machine learning methodologies.

Potential areas for added functionality

Areas of functionality that may potentially be addressed further include dealing with concurrency issues. As of right now, the database essentially functions with only one user able to use it at a time. On top of that, there is additional backup and recovery functionality that can be improved within the database.

In addition, this project can be developed further by implementing a visual GUI, like a mobile or web application or a website. Building a front-end side to our project would make it more approachable, understandable, and usable to users.

One other potential area for added functionality may be expanding the processes able to be done with the database. For example, added functionality could come in the form of being able to create more complex read queries that allow you to search up multiple classes and phylums at once or even being able to delete your account or recover a confirmation code.