The Pokemon Database Experience Report

COMP 378 Database Systems

10 May 2022

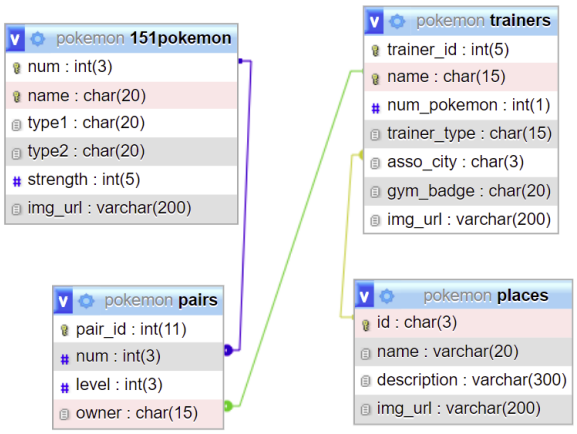
Leah Goldberg & John Chu

Project GitHub: https://github.com/JuheonChu/pokemonDB

The Database

General Design

We have created a relational database based off of the fictional universe of *Pokemon*. The overall design process, including the creation of the tables went well and was enjoyable. Our final design consists of 4 tables: *151Pokemon* (151 rows), *Trainers* (15 rows), *Pairs* (65 rows), and *Places* (10 rows). These tables contain information about Pokemon, Pokemon trainers, Pokemon and trainer pairs, and places found throughout the Pokemon universe. The diagram below provides a visual on how the tables are set up and connected via foreign keys.



Relation Notation

The section below notates the Pokemon database schema in relation notation. Primary keys are underlined and foreign keys are *italicized*.

POKEMON (num, name, type1, type2, strength, img_url)

TRAINER (trainer_id, name, num_pokemon, trainer_type, *asso_city*, gym_badge, img_url)

asso_city: foreign key, refers to ID in PLACE, NULL value not allowed

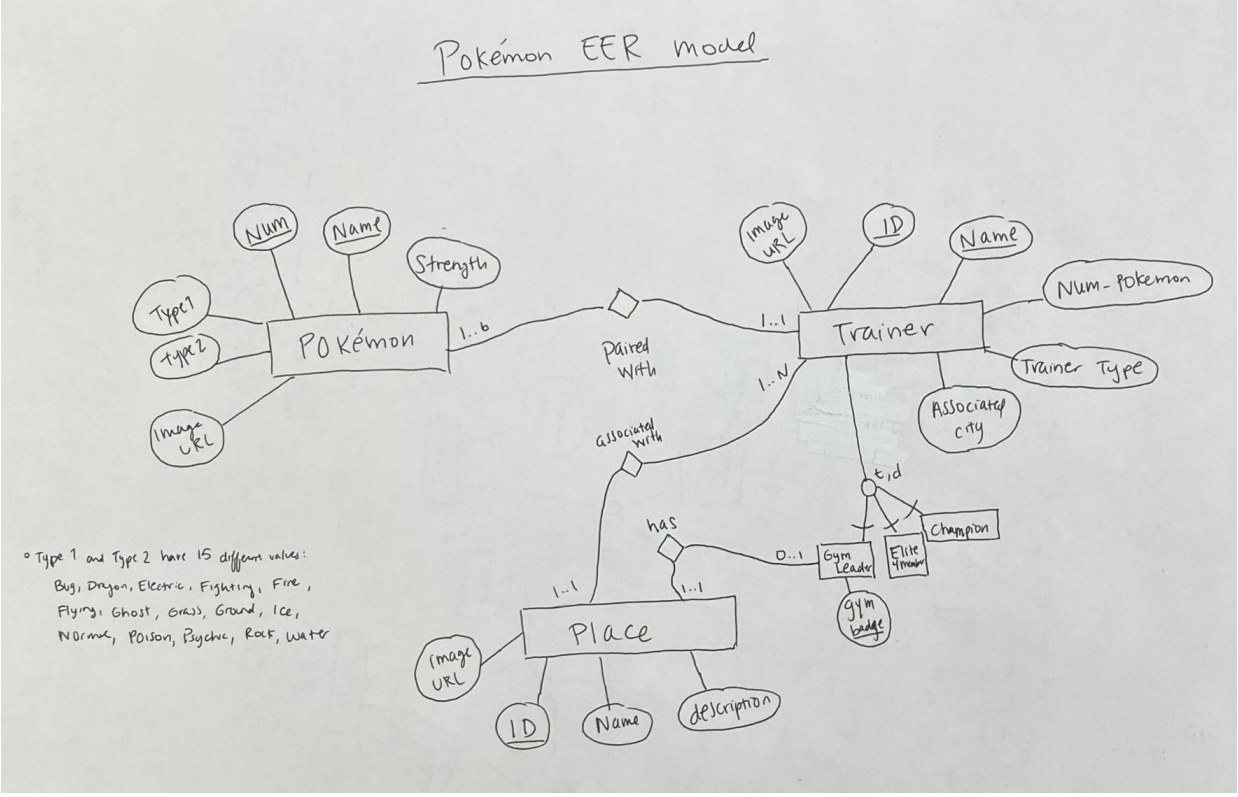
PAIRS(pair_id, *num*, level, *owner*)

num: foreign key, refers to NUM in POKEMON, NULL value not allowed

owner: foreign key, refers to NAME in TRAINER, NULL value not allowed

PLACE(id, name, description, img_url)

Enhanced Entity Relation Model



The Computer Program

Program Description & Functionality

We have created a dynamic website that runs on JSP (Java Server Page), and connects to the tables in Pokemon database schema via JDBC and MySQL workbench. This project is a MVC structured model. MVC is an acronym standing for Model, Viewer, and Controller. Model is a part where we use program logic. We use Data Transfer Objects and Data Access Objects in order to complete the program functionalities. Note that Data Transfer Object stands for each table in the Pokemon schema, and Data Access Object represents the JDBC object to connect to our database server. The Controller serves as an arbiter between the Viewer and Controller by mapping the requested URL address from the Viewer and forwarding it to the Model part. The website has three main pages: Pokemon, Trainers, and Places, and a search functionality. All of these pages display images that correlate with its information and each page has slightly different features.

Pokemon: This page displays all of the information from the *151pokemon* table. The user can also sort the Pokemon on this page by number (1), name (2), and strength (3 & 4). The queries used for the filter functionality are as follows:

```
1. String sql = "SELECT * FROM pokemon.151pokemon"
```

2. `String sql = "SELECT * FROM pokemon.151pokemon pok ORDER BY pok.name ASC";`
3. `String sql = "SELECT * FROM pokemon.151pokemon pok ORDER by pok.strength DESC";`
4. `String sql = "SELECT * FROM pokemon.151pokemon pok ORDER by pok.strength ASC";`

Pairs: This page contains information on each trainer from the *Trainers* table, along with the Pokemon that each trainer owns, which is information from the *Pairs* table. A JOIN query that joins *Trainers, Pairs,* and *151Pokemon* is used to display all of the information on this page. The query is as follows:

```
String sql = "SELECT\r\n"
        + "  pair.owner, tr.type, pair.level, pok.name, pok.strength, "
        + "  tr.num_pokemon, pok.img, tr.image, pok.type1, pok.type2, "
        + "  pok.color1, pok.color2\r\n"
        + "FROM\r\n"
        + "    pokemon.pairs pair\r\n"
        + "Left JOIN pokemon.151pokemon pok ON\r\n"
        + "    pair.num = pok.idx\r\n"
        + "Left JOIN `pokemon`.trainers tr ON\r\n"
        + "    pair.`owner` = tr.`name`\r\n"
        + "ORDER BY\r\n"
        + "    tr.name\r\n"
        + "";
```

Places: This page contains the name, description, place ID, and images from each of the places in the *Places* table. The query used to display information for this page is as follows:

```
String sql = "SELECT * FROM pokemon.places";
```

The Search Functionality

The top of the website also includes a search bar. With this, the user can search up any Pokemon, Trainer, or Place. While searching, the user can search any part of the string (of a name, type, description, etc.) and results that match the string will be returned. This feature is implemented by using the keyword 'LIKE' in SQL queries. The following SQL queries are used to retrieve this information:

Pokemon (can be searched by Name or Type):

```
String sql = "SELECT * FROM pokemon.151pokemon WHERE `name` LIKE ?"
        + "OR `type1` LIKE ? OR `type2` LIKE ?";
```

Trainers (can be searched by Name or Trainer Type):

```
String sql = "SELECT * FROM pokemon.trainers WHERE `name` LIKE ? OR `type` LIKE ?";
```

Places (can be search by Name or Description):

```
String sql = "SELECT * FROM pokemon.places WHERE `name` LIKE ?"
        + "OR `description` LIKE ?";
```

Instructions for running the server

1. Ensure the Driver connection is set up properly in all of the DAO files
2. Go to `src` → `main` → `webapp` → `WebContent` → `index.jsp`
3. Right click in `index.jsp`, then click "*Run As*" → "*Run on server (Tomcat 8)*"

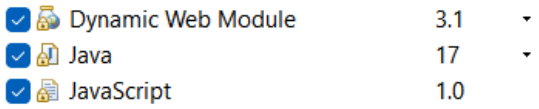
Technical Database Notes

- Data Access Object (DAO) files handle all the tasks related to extracting the data from the database server using query statements.
- Data Transfer Object (DTO) files are used to store the data extracted from the database server.
- Prepared statements are used in all of the DAO files to improve the functionality of the Statement object. When searching the information of Pokemon, places, and trainers, searched words can be set as parametric strings in the PreparedStatement object and store the information of each tuple into the Data Transfer Object.
- In the 151Pokemon table, there are two extra columns called "color1" and "color2" that correspond to the colors of "Type1" and "Type2" when they are visually displayed on the web page via the CSS. These colors have hexadecimal RGB values.

Technical Programming Notes

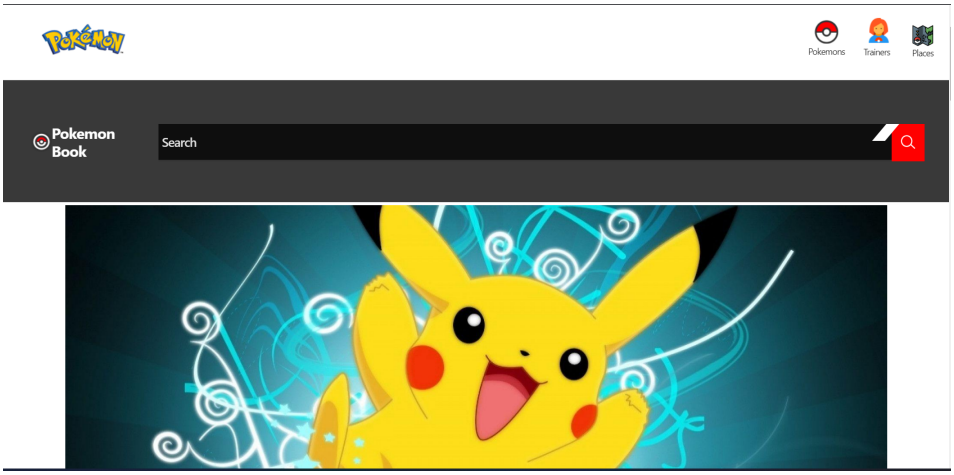
- The Pokemon project is a JSP based MVC model.
- In this project, we are using servlet as a controller, JSP as a viewer component, and the Java class as a model. The functionalities of the model, viewer, and controller are provided as follows:
    - Model: The model is a chief component of MVC pattern. We applied dynamic data structures and managed data, program logic on the model part. In our Pokemon project, all the action objects from the action factory have all the program logic using the DAO and DTO objects.
    - Viewer: The viewer simply represents the component that can be visualized on screen. We used JSP files to display all the required information for each component of the program functionalities.
    - Controller: The controller prepares the logics mapped by the requested data through the queries and selects the viewer name to complete the user-request.

- We are using following versions to enable our JSP projects:

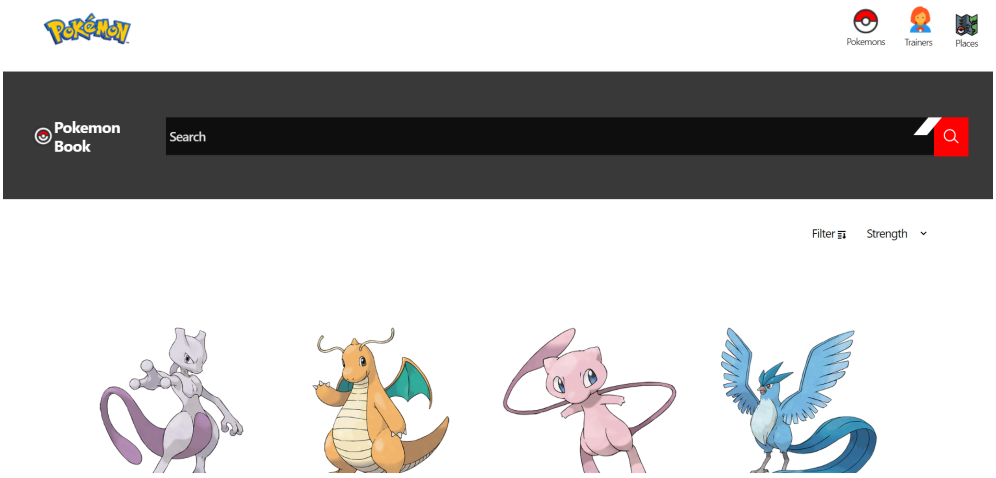


Sample screenshots of the server web

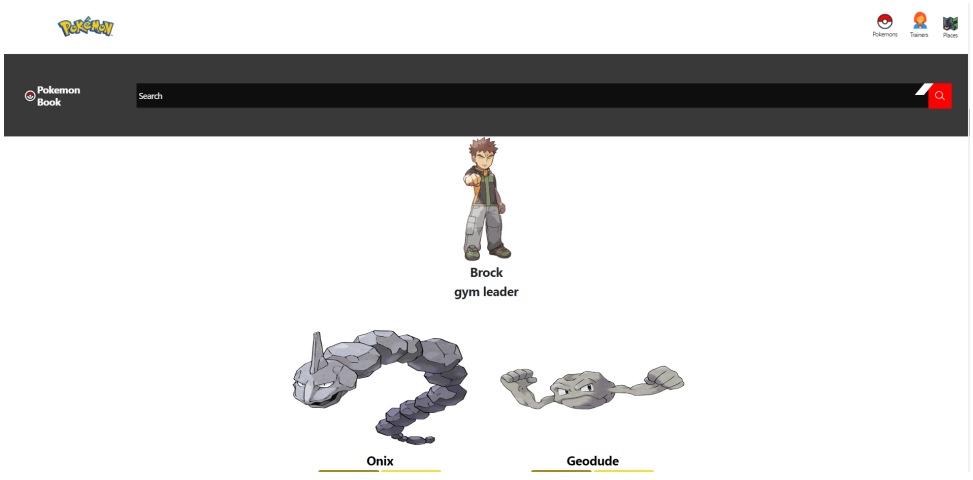
Main Page



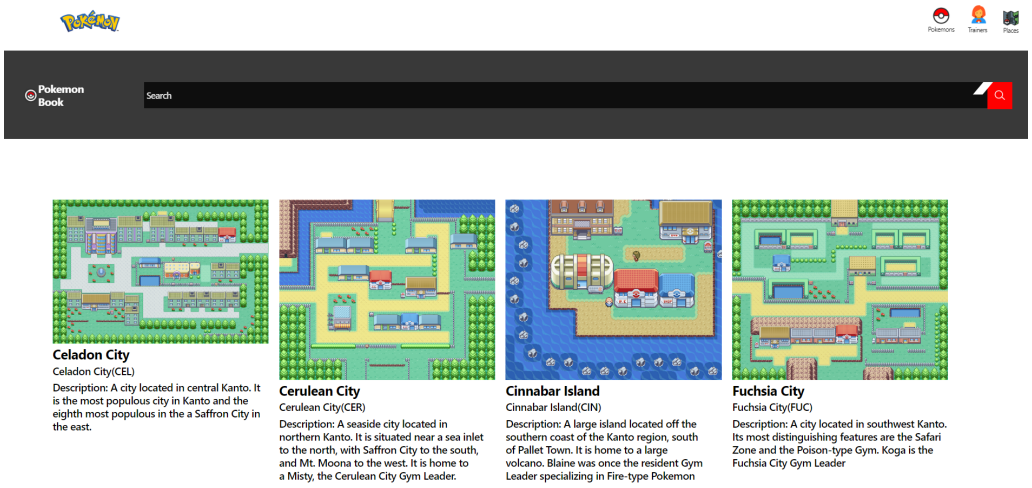
Pokemon Page



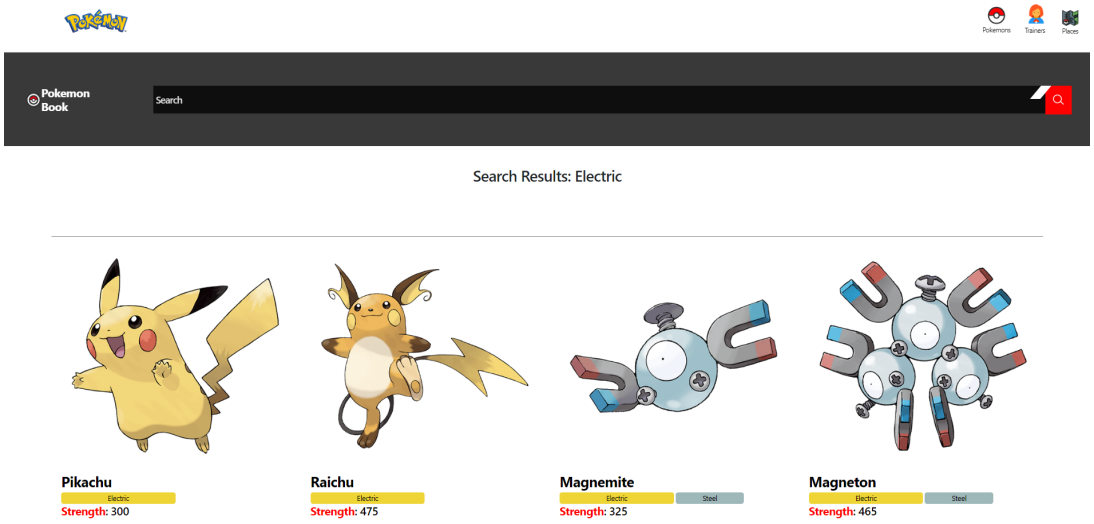
Trainer Page



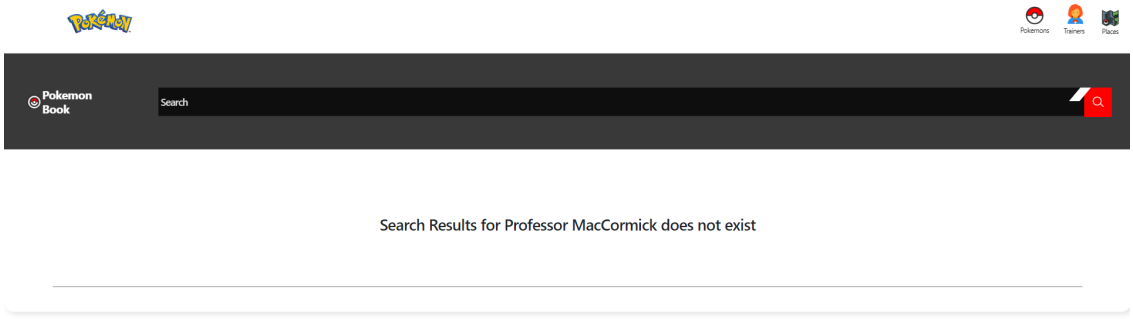
Place Page



Search Results Page



No Results Page



Overall Experience

General Thoughts

Overall, we enjoyed the creativity and innovation of this project greatly. It has been a valuable learning experience for us to construct a database system that interacts with the dynamic web page. Moreover, we were motivated by our passion for Pokemon to complete this project enthusiastically and successfully.

Difficulties

Creating the database was a fairly straightforward process. It took a while to come up with a final design, but the process itself was enjoyable. We also did not have too much trouble creating any of the queries needed for the web pages or search functionality. Most of our difficulties were not related to the database aspect of the project, but with the CSS aspect and rendering the pages properly. Furthermore, the JSP project necessitates numerous configurations, including Project Facet, Tomcat, STS tools, and JDBC, which were a challenge to set up. Additionally, because each dynamic web page in our project combines components from HTML/CSS, Javascript, JQuery, JSTL, Java, and JSP, it was difficult to determine where we should begin debugging to troubleshoot our programs.

Future Implementations

Additional Features

The following features can be added to improve our functionality of Pokemon project with the goal of making our web page more community minded:

1. Expanding the Pokemon Database: There are currently around 900 Pokemon, and many more trainers and places than the ones listed in our database. We could expand our database and implement more pages on our web page to reflect all of this information.
2. Pokemon Blog: We can potentially make a blog where people can feel free to share their feedback on Pokemon by allowing people to write the blog post (file or picture included), leave comments, and delete the posts that they wrote.
3. Strengthening searching: We can potentially strengthen the searching functionality by producing all the Pokemon owned by the searched trainer. Similarly, we can further reinforce our searching by producing all the trainers that are associated with the places.
4. Commercializing: We can also improve our functionality by adding a page where users can buy Pokemon products, such commercialized goods for each Pokemon and trainer. Specifically, we can implement database transactions when the customer clicks the “purchase” button, and the paid bill can be added into the bank account of an administrator. Note that we will assume that the total monetary amount of the user is going to be subtracted by the amount of the bill that the user paid for security-purpose.

5. Admin page: We can further make a page for the administrative users and regular users to separate the functionality depending on who uses this web page. Specifically, we will start by implementing the membership functionality, such as log-in, sign-up, and finding ID and password. Then, each user will have his/her type: admin or regular user. Then, regular users can buy the Pokemon goods, search Pokemons, and visualize all the Pokemons, trainers, and places. The administrative users can manage his/her annual outcome, visualize the feedback of the customers on each Pokemon goods.

This concludes our report on the Pokemon Database Project.