Team Saltine Crackerines
Connor Bennudriti
Brinley Hull
EECS 447 Databases
Project Final Report
5-10-24

Website: www.saltine.wuaze.com

Introduction

We are managing a Pokemon database with information about each Pokemon, their types, and their players. It's important to keep track of the Pokemon information so that we can know things like which Pokemon belong to which players, what types belong to which Pokemon, etc. A database will also make inserting, removing, and updating each Pokemon from each corresponding table (such as adding a Pokemon to a team, updating a Pokemon weight, etc.) easy. The main purpose of our project is to allow players to add Pokemon to their teams.

Our website allows users to create a player or log in to an existing player and filter Pokemon based on the Pokemon's attributes. Once the player has found Pokemon that they would like to add to their team, they can select or deselect it to add or remove it from their team. Finally, we allow the user to see their selected pokemon on a special team view page on the site. For a detailed guide through our website's functionality, please view our presentation: https://www.youtube.com/watch?v=DYOBGazjkSg&t=13s&ab_channel=BrinleyHull

We host our website on the infinityfree hosting server. The relevant Pokemon tables are held in the hosting server's database. The name of our website is www.saltine.wuaze.com. Description of our tables and queries used are detailed in later sections. However, if access is needed to the database tables, please visit https://dash.infinityfree.com/accounts/if0_36325610/databases and use the following login information:

email: brinleyhull53@ku.edu
password: Crackerines12!

Requirements Analysis

Data:

1. Pokemon: number, name, type1, type2, hp, attack, defense, special attack, special defense, speed
2. Player: name, color
3. Types: name, strengths, weaknesses (can be derived, so not stored)

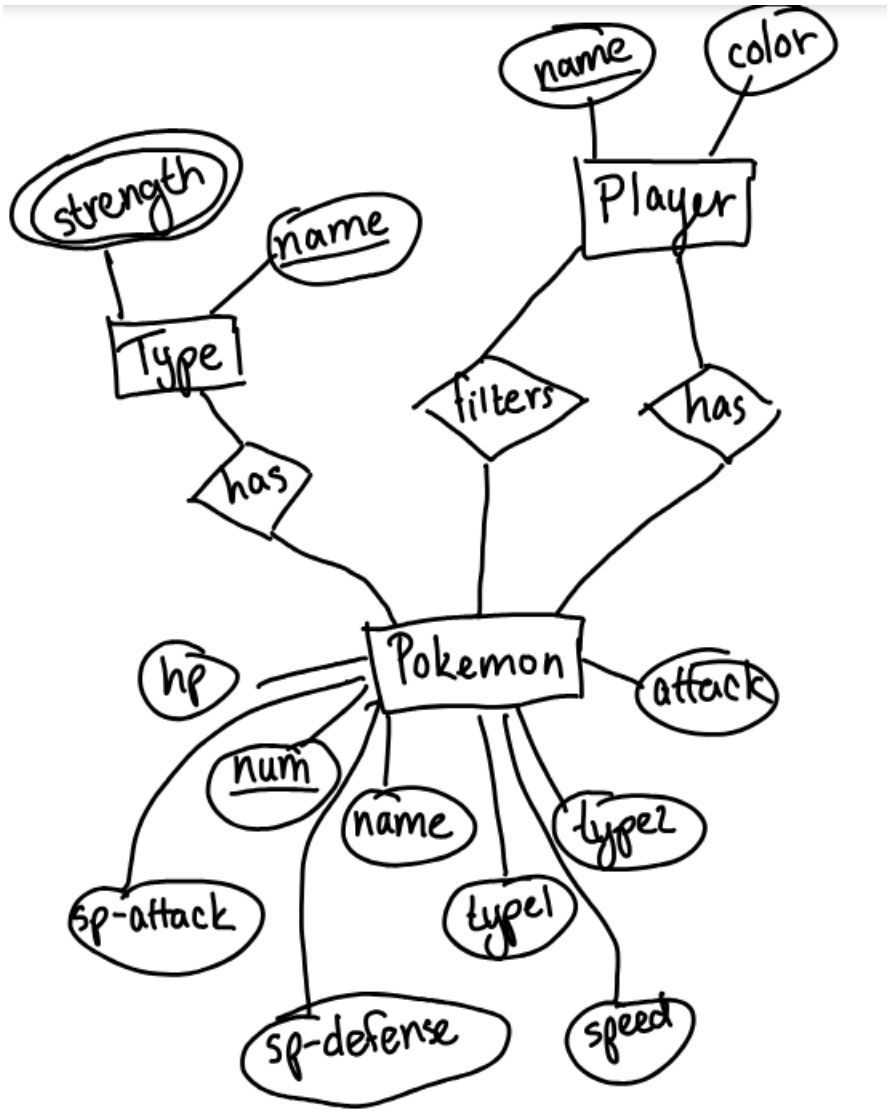
Constraints:

1. Player name must be unique
2. Pokemon name must be unique
3. Types can have multiple strengths
4. A player has a team color
5. A player can filter Pokemon
6. Domain Constraints:
    a. Player
        i. name: string (key)
        ii. team_color: string
    b. Pokemon
        i. poke_num: int > 0 (key)
        ii. name: string
        iii. type1: string
        iv. type2: string
        v. hp: int
        vi. attack: int
        vii. defense: int
        viii. special attack: int
        ix. special defense: int
        x. speed: int
    c. Types
        i. type_name: string (key)
        ii. strong_against: string

Operations:

1. Adding/removing Pokemon to a player
2. Adding entries in Player

System Architecture: ER Diagram



Constraints:

1. Pokemon have hp, attack, defense, special attack, special defense, speed, name, and are identified by number.
2. Types have strengths and are identified by name.
3. Players have a team color and are identified by their name

System Architecture: Tables/Relational Schema

Player(name, color)
Pokemon(num, name, type1, type2, hp, attack, defense, sp-attack, sp-defense, speed)
Pokemon_in_Team(player_name, pokemon_num)
Type(name)
Type_Strength(type_name, type_strength)
Filtered(pokemon_num, player_name)

The tables that the user is able to perform actions on are Player (add entries), Pokemon_in_Team (add and remove entries), and Filtered (add and remove entries when searching for Pokemon). The filtered table is used to keep track of the Pokemon being filtered by the user. The other three tables above are static and simply hold information about entities.

Implementation and Queries Used

We implemented our website using PHP, MySQL, HTML, CSS, and JS on the infinityfree hosting server. We kept track of current user using the SESSION variable. The main two files for our code are the new_team_creator.php and searchpokemon.php.

Generic Queries Used:

- SELECT Name FROM Player
- SET SQL_BIG_SELECTS=1

Dynamic Queries Used:

- INSERT INTO Player VALUES ("' . $name . '", "' . $color . '")
- DELETE FROM Pokemon_In_Team WHERE Player_Name = '" . $_SESSION['user'] . "' AND Pokemon_ID =" . $row['ID'] . "
- SELECT ID FROM Pokemon WHERE Name='" . $_GET['id'] . "'
- INSERT INTO Pokemon_In_Team VALUES ('" . $_SESSION['user'] . "', " . $row['ID'] . ")
- SELECT * FROM Player WHERE Name="' . $name . '"
- SELECT DISTINCT LOWER(Name) FROM Pokemon_In_Team, Pokemon WHERE Player_Name='" . $_SESSION['user'] . "' AND Pokemon_ID=ID AND ID>0
- SELECT DISTINCT LOWER(Pokemon.Name) FROM Filtered, Pokemon WHERE ID=Filtered.Poke_Num AND Player='" . $_SESSION['user'] . "'
- DELETE FROM Filtered WHERE Player="' . $_SESSION['user'] . '"
- Additional query described below

Queries with Join Used:

- SELECT DISTINCT LOWER(Name) FROM Pokemon_In_Team, Pokemon WHERE Player_Name='" . $_SESSION['user'] . "' AND Pokemon_ID=ID AND ID>0

- SELECT DISTINCT LOWER(Pokemon.Name) FROM Filtered, Pokemon WHERE ID=Filtered.Poke_Num AND Player='" . $_SESSION['user'] . "'
- Additional query described below

Complicated dynamic/join query:

Our code also contains a complicated selection and insertion query that makes use of subqueries, joins, and dynamic elements depending on which filtering criteria the user creates. The query makes up a large portion of searchpokemon.php code file (variable name $query). An important feature to note is how it dynamically creates a self-join for table Type_Strength depending on how many checkboxes the user has selected. The code copied below highlights one of the instances of this phenomenon in the weakness query, adding a join for however many checkboxes selected (kept track of with variable $count).

```
$wq .= " FROM Type_Strength";
for ($i = 0; $i < $count-1; $i++) {
    $wq .= ", Type_Strength v" . $i;
}
```