Pokémon Database Final Report

by

Richard Anderson, Dakota Degidio, Lucas Martin, Mike Moses, Preston Mouw

Project Description

Our project is a relational database containing information on the various Pokémon. Pokémon is a popular franchise centered on fictional creatures, which humans try to train and catch for their unique abilities. It is a large franchise consisting of multiple movies, TV series, and video games. At the time of this writing, there are over 800 Pokémon, each of which have many traits and attributes to be stored in a database.

Because of the sheer volume of characters and their overall complexity, it is highly useful for avid Pokémon fans and gamers to be able to access information from an online database. The motivation of this project is to provide easy access to Pokémon-related information for fans of the series. In short, it is something that we enjoy. Our targeted clients are simply fans of the series that want to search the web for information on their favorite Pokémon. Our clients will be able to search for information using a web interface, which is detailed in this proposal. We plan to publish the database online.

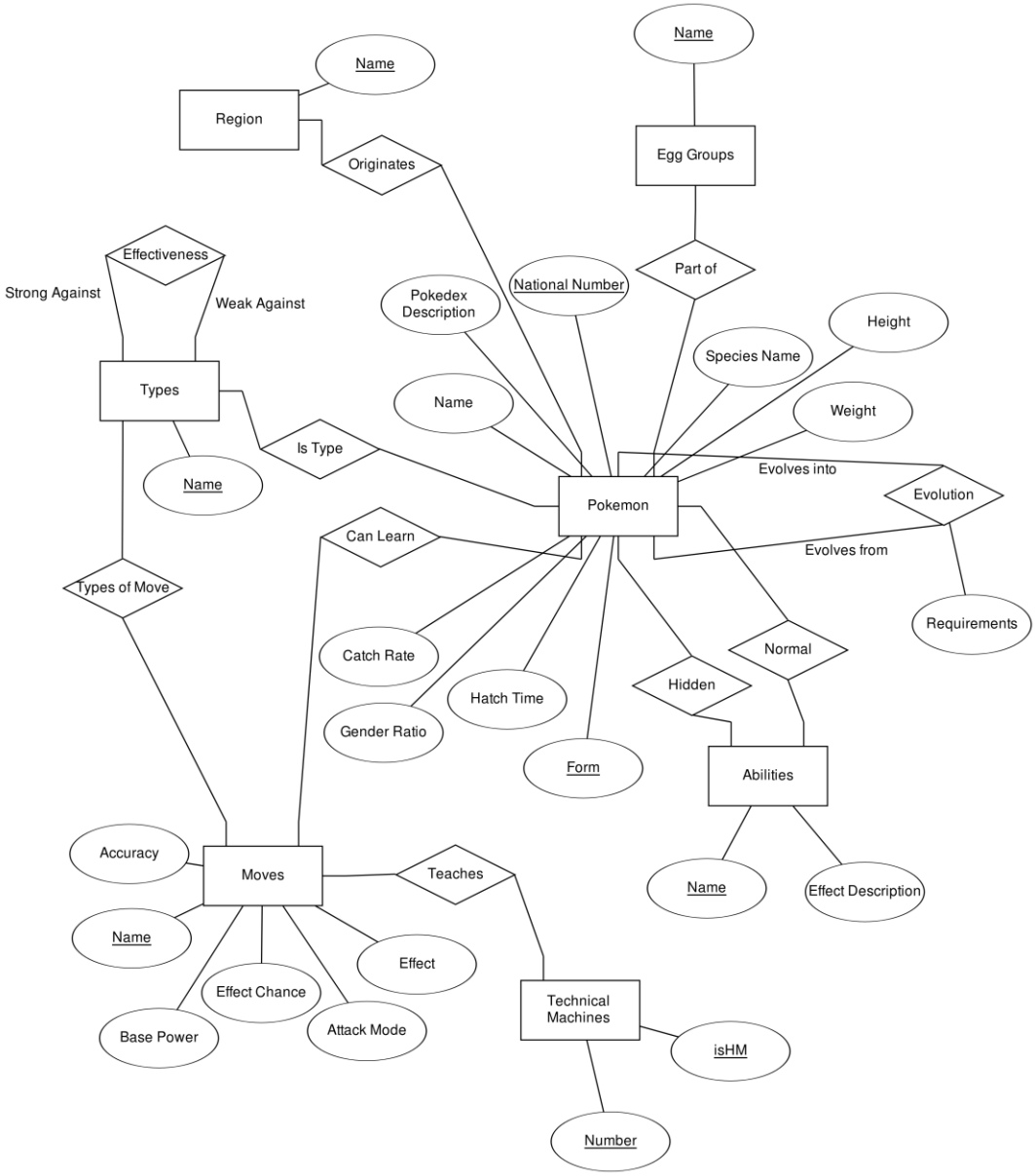
Requirements

Our database keeps track of Pokémon statistics. Operations that our database must support include searching for Pokémon, regions, moves, abilities, and types, and supplying all the attributes that they contain. These links must go both ways for streamline searches starting at any identifying entity or relationship. We want to make sure there isn't a struggle for the user to find something especially if they only know a part of the exact Pokémon.

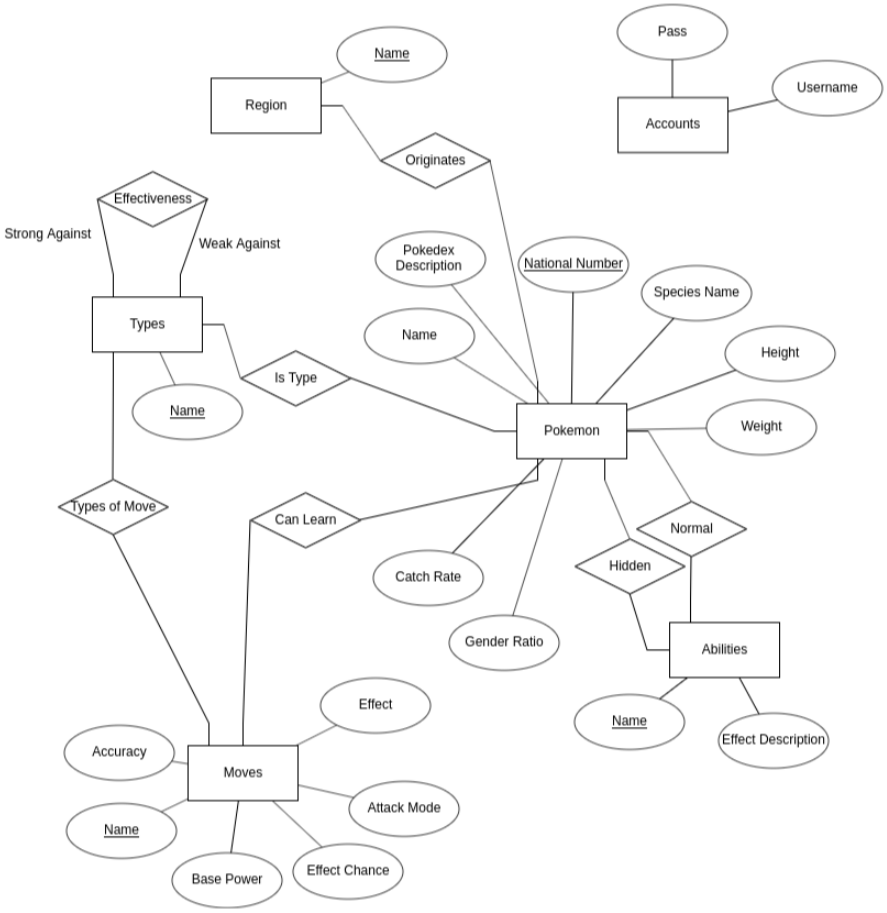
Specifically, if a user knows only the type of a Pokémon that they can search by type instead of by name and narrow down their search as they go. We also want the user to not be limited by the database's total Pokémon data allowing the user to add/create a Pokémon of their own and select their attributes from the existing list.

Website administrators have the same capabilities as users. Assumptions for efficient usage would require the user to know the name of the Pokémon they are searching for. This would be the fastest way to get to the statistics of that Pokémon. Otherwise they would need to search through other entities to eventually get to the original Pokémon they were looking for.

Starting E/R Diagram



Final E/R Diagram



Normalized Database Schema 3NF

NatNum_Pokemon (nat_num, gender_ratio, species_name, pokedex_desc, catch_rate, img)

Types (name)

Strong_Against (attacking, defending, foreign keys (attacking, defending) references Types(name))

Moves (name, effect, effect_chance, base_power, attack_mode, accuracy, movetype, foreign keys(movetype) references Types(name))

IsType (nat number, type, foreign keys(nat number) references Pokemon(national number), foreign keys(type) references Type(name))

Can_Learn (nat-num, move-name, foreign keys(nat num) references Pokemon(national number), foreign keys(move name) references Moves(name))

Abilities (name, effect desc.)

Normal_Abilities (nat num, ability name, foreign keys(nat num) references Pokemon(nat num, form), foreign keys(ability name) references Abilities(name))

Hidden_Abilities (nat num, ability name, foreign keys(nat num) references Form_Pokemon(nat num, form) foreign keys(ability name) references Abilities(name))

Region (name, generation,description, img)

Originates (nat num, region, foreign keys(nat num) references Pokemon(nat num,) foreign keys(region) references Region(name))

Accounts (username, hash_pass)

Components of our Implementation

Our implementation was mostly centered around creating questions or problems then finding the most feasible solution to them. The first main component was searching for Pokémon, Regions, Moves, and Abilities by name. This was done by creating PHP pages and linking forms that queried our MySQL database. Each of these categories also have their own main page listings which can be sorted by different attributes as well as filtered by specific attributes. Any user who accesses our website has editing permissions such that they can add or edit any of the tuples, however, they cannot delete things. This requires Admin login access which is implemented by a tuple of usernames and hashed passwords.

http://acadweb1.salisbury.edu/~lmartin9/386project/index.php

Functionalities

- **Searching --** Users can search Pokémon, regions, moves, abilities, and types
- **Insertion –** Users can add Pokémon, regions, moves, abilities, and types
- **Edit** (not delete) -- Users can modify entries for Pokémon, regions, moves, abilities, and types
- **Delete** (Admin only) -- Users with administrative privileges can remove entries from the database application

Software/Platform/Environment

The main platforms we used to implement our database application were Github, MySQL, and Acadweb1. Github provided easy version control, which was a great help because the website was being hosted on just one of our Acadweb1 accounts so it would be difficult to upload different parts of the project from multiple people working simultaneously. By cloning a Github repository to Acadweb1, we were able to upload things to Github from separate accounts and then all the project lead had to do was git pull from their Acadweb1 account and it would update the website on the server. Additionally, this made connecting the MySQL database relatively easy.

Testing

Our application was tested by going through each page on the live website and going through the various links making sure that they went to the right places. After checking everything worked correctly for the expected results, we then tried to break the application by testing extraneous inputs making sure there were not any undefined results.

Test 1: proper insertion of new data

- Using our newly created web forms, we tested out the "insert" functionality
- First, we check the displayed/updated webpage to see if it was consistent with the information we inserted
- Next, for the sake of being thorough we viewed the SQL tables from the Linux command line to ensure that everything was proper

Test 2: Deletion

- Delete an entry using the web interface, (assuming we have admin privileges). We made sure that the corresponding information disappeared from the tables on the web page.
- Similar to test 1, we verified that the information no longer exists in the SQL table from the Linux command line interface

Test 3: Edit buttons

- For each edit button, we made basic changes, then double checked the SQL tables reflected those changes

Teamwork and Learning

Our project relied heavily on each of our members participating in our meetings and completing our assigned goals every week or biweekly, (depending on the task). We agreed to meet at least one day a week to go over what each member had completed for their assigned task. Since we were only meeting one day a week, we formed a group chat so that we could keep in contact regularly throughout the week. This allowed us to collaborate on our project and solve any minor/major issues in our database as a team. This ensured that nobody was left out of the learning process when implementing our database.