Katie Chang

Alec Tabatchnick

Databases 600.315

Application Domain

Application is a stand-alone Pokemon game database containing information that helps players to look up relevant facts and answers. The application answers types of questions in order to enhance the playing experience and help the player to try to catch them all.

Database Final Project Phase 2

1. **Edited Queries** 
   1. How many different Pokemon have a given ability, grouped by generation?
   2. In which region(s) can I catch a given Pokemon?
   3. In which location(s) can I catch a given Pokemon?
   4. What Pokemon can I catch at a given location?
   5. Which Pokemon from a given generation has the best total base stats?
   6. Select all Pokemon that is of two given types.
   7. Select all Pokemon with more than one evolution that are of one given type
   8. Select the name, type, and weight of pokemon who weigh more than a given weight and are of a given type.
   9. Select the types of all pokemon who can use a given move.
   10. Select all pokemon who can use a given move.
   11. Show number of unique pokemon for a given generation.
   12. Show number of unique pokemon in a given type.
   13. Select pokemon that is of one given type.
   14. Select the pokemons that require a held item in order to evolve.
   15. Select the pokemons that require a trigger item in order to evolve.
   16. For a given pokemon, list its evolution chain / all its evolved / pre-evolved forms.

*Queries were edited from Phase 1 because of data set limitations (we made those queries before knowing fully what the dataset contained.)*

**2. How Database was loaded**

We found csv files from this source: <https://github.com/veekun/pokedex/tree/master/pokedex/data/csv>

We processed the data if necessary using a Java scripts that can be found in the “processing” directory, then used <http://sqlizer.io/> to turn the CSV into SQL table definitions and insert statements.

We created a new database on mysql called ‘pokedex’.

We then ran those SQL definitions to populate the database. This whole SQL file is located in pokedexFinal/sql\_commands/all\_files.sql

Please note to change the mysql location, user, and password to yours in AppService.java and Bootstrap.java.

**3. Platform**

Built on a Windows Surface Book running a Virtual Machine Ubuntu with a local mysql database.

**4. Brief User guide**

Clone or download the zip from: **https://github.com/batchii/pokedexFinal**

You will need mysql, Java and Maven.

To create the Pokedex and populate it, run

create database pokedex;

.\ pokedexFinal/sql\_commands/all\_files.sql;

To load the stored procedures, run

.\ pokedexFinal/sql\_storedProcedures\_rd.sql

To build the project, run

mvn clean install from the main directory

To run the build, run

java -jar target/FinalProj-1.0-SNAPSHOT.jar also from the main directory

To access the web application, go to

<http://localhost:8080/> on your favorite browser.

From there, have fun!

**5. Major/minor areas of specialization**

Stored Procedures: Used for every question / query in the web application. Each of these stored procedures has one job and was made to enable the user to search for things he/she may be interested in.

-We created a bootstrapped web interface to enable easy navigation of our queries. The app uses JQuery.js to manipulate the web page so that the page does not reload on each request, instead just updating the current page. The app also uses a Spark Java Framework server to allow the app to run continuously and to expose REST endpoints to allow users to query the server and transitively, the database. This provides a safe interface for users, and prevents them from seeing the contents of the database or how it is called.

-Data curation with scripting: In order to upload our database we had to make sure that the csv files we used had the material we needed, and eliminated additional details that may have been unnecessary for our goals. This was done in Java with a few scripts that will curate the code.

**6. Analysis of system’s limitations / List suggested possibilities for improvement**

The system is only limited to a few queries currently and we do not offer word prediction as an option. In the future this may help users of our app to find the pokemon, or things they want to search for. Additionally, we do not offer very much flexibility in our queries as they are all pre created as procedures. The frontend does not have very good error handling right now as well.

**7. Sample Output**

\*Please take a look at the folder pokedexFinal/submissionImages to see screenshots of the resulting front-end development for our Pokedex.

The following are the results of the written stored procedures that this web application generates.

1. How many different Pokemon have a given ability, grouped by generation?
2. Typing in ‘levitate’ will give you
3. +---------------+------------+-----------+
4. | generation\_id | identifier | numUnique |
5. +---------------+------------+-----------+
6. | 1 | levitate | 5 |
7. | 2 | levitate | 2 |
8. | 3 | levitate | 10 |
9. | 4 | levitate | 10 |
10. | 5 | levitate | 5 |
11. +---------------+------------+-----------+
12. In which region(s) can I catch a given Pokemon?
    1. try typing ‘tentacool’ -> will result in a table with these entries
    2. +-----------+--------+
    3. | Pokemon | Region |
    4. +-----------+--------+
    5. | tentacool | hoenn |
    6. | tentacool | johto |
    7. | tentacool | kanto |
    8. | tentacool | sinnoh |
    9. | tentacool | unova |
    10. +-----------+--------+
13. In which location(s) can I catch a given Pokemon?
    1. Typing in ‘pikachu’ gives you
    2. +---------+------------------+
    3. | Pokemon | Location |
    4. +---------+------------------+
    5. | pikachu | hoenn-route-109 |
    6. | pikachu | johto-route-38 |
    7. | pikachu | lilycove-city |
    8. | pikachu | mirage-tower |
    9. | pikachu | mossdeep-city |
    10. | pikachu | pc-nagoya |
    11. | pikachu | pokemon-event-10 |
    12. +---------+------------------+
14. What Pokemon can I catch at a given location?
    1. Typing in ‘canalave-city’ gives you
    2. +-----+------------+
    3. | id | identifier |
    4. +-----+------------+
    5. | 72 | tentacool |
    6. | 73 | tentacruel |
    7. | 120 | staryu |
    8. | 129 | magikarp |
    9. | 130 | gyarados |
    10. | 278 | wingull |
    11. | 279 | pelipper |
    12. | 422 | shellos |
    13. | 423 | gastrodon |
    14. | 456 | finneon |
    15. | 457 | lumineon |
    16. +-----+------------+
15. Which Pokemon from a given generation has the best total base stats?
    1. typing in ‘1’ (generations go from 1 to 6)
    2. +---------+---------------+
    3. | Pokemon | TotalBaseStat |
    4. +---------+---------------+
    5. | mewtwo | 680 |
    6. +---------+---------------+
16. Select all Pokemon that is of two given types.
    1. Typing in ‘fire’ and ‘ground’ gives you
    2. +-------+----------------+---------+---------+
    3. | id | identifier | TypeOne | TypeTwo |
    4. +-------+----------------+---------+---------+
    5. | 322 | numel | ground | fire |
    6. | 323 | camerupt | ground | fire |
    7. | 10078 | groudon-primal | ground | fire |
    8. | 10087 | camerupt-mega | ground | fire |
    9. +-------+----------------+---------+---------+
17. Select all Pokemon with more than one evolution that are of one given type
    1. Typing in ‘ghost’ will give you
    2. +------------+------------+
    3. | pokemon | typeOne |
    4. +------------+------------+
    5. | haunter | ghost |
    6. | gengar | ghost |
    7. | shedinja | ghost |
    8. | banette | ghost |
    9. | dusclops | ghost |
    10. | drifblim | ghost |
    11. | mismagius | ghost |
    12. | dusknoir | ghost |
    13. | froslass | ghost |
    14. | cofagrigus | ghost |
    15. | jellicent | ghost |
    16. | lampent | ghost |
    17. | chandelure | ghost |
    18. | golurk | ghost |
    19. | doublade | ghost |
    20. | aegislash | ghost |
    21. | trevenant | ghost |
    22. | gourgeist | ghost |
    23. +------------+------------+
18. Select the name, type, and weight of pokemon who weigh more than a given weight and are of a given type.
    1. Typing in ‘1000’ and ‘fire’
    2. +-------+------------------+------+--------+
    3. | id | Pokemon | Type | weight |
    4. +-------+------------------+------+--------+
    5. | 59 | arcanine | fire | 1550 |
    6. | 244 | entei | fire | 1980 |
    7. | 250 | ho-oh | fire | 1990 |
    8. | 323 | camerupt | fire | 2200 |
    9. | 485 | heatran | fire | 4300 |
    10. | 500 | emboar | fire | 1500 |
    11. | 643 | reshiram | fire | 3300 |
    12. | 721 | volcanion | fire | 1950 |
    13. | 10034 | charizard-mega-x | fire | 1105 |
    14. | 10035 | charizard-mega-y | fire | 1005 |
    15. | 10078 | groudon-primal | fire | 9997 |
    16. | 10087 | camerupt-mega | fire | 3205 |
    17. +-------+------------------+------+--------+
19. Select the types of all pokemon who can use a given move.
    1. Typing in ‘tackle’ gives you
    2. +------------+
    3. | identifier |
    4. +------------+
    5. | bug |
    6. | dark |
    7. | dragon |
    8. | electric |
    9. | fairy |
    10. | fighting |
    11. | fire |
    12. | flying |
    13. | ghost |
    14. | grass |
    15. | ground |
    16. | ice |
    17. | normal |
    18. | poison |
    19. | psychic |
    20. | rock |
    21. | steel |
    22. | water |
    23. +------------+
20. Select all pokemon who can use a given move.
    1. Typing in ‘oblivion-wing’ will give you
    2. +---------+
    3. | Pokemon |
    4. +---------+
    5. | yveltal |
    6. +---------+
21. Show number of unique pokemon for a given generation.
    1. Typing in ‘5’ gives you
    2. +----+--------------+---------------+
    3. | id | identifier | UniquePokemon |
    4. +----+--------------+---------------+
    5. | 5 | generation-v | 156 |
    6. +----+--------------+---------------+
22. Show number of unique pokemon in a given type.
    1. Typing in ‘water’ gives you, other options include ground, fire, electric… etc
    2. +------------+---------------+
    3. | SearchType | UniquePokemon |
    4. +------------+---------------+
    5. | water | 128 |
    6. +------------+---------------+
23. Select pokemon that is of one given type.
    1. Typing in ‘water’ gives you
    2. +-------+--------------------+-------+
    3. | id | pokemon | type |
    4. +-------+--------------------+-------+
    5. | 7 | squirtle | water |
    6. | 8 | wartortle | water |
    7. | 9 | blastoise | water |
    8. .........
    9. ……..
24. Select the pokemons that require a held item in order to evolve.
    1. Typing in ‘protector’ (which is an item) gives you
    2. +-------------+-------------+
    3. | Pokemon | HeldItem |
    4. +-------------+-------------+
    5. | rhydon | protector |
    6. +-------------+-------------+
25. Select the pokemons that require a trigger item in order to evolve.
    1. Typing in ‘thunder-stone’ gives you
    2. +-------------------+--------------------+
    3. | Pokemon | TriggerItem |
    4. +-------------------+--------------------+
    5. | pikachu | thunder-stone |
    6. | eevee | thunder-stone |
    7. | eelektrik | thunder-stone |
    8. | pikachu-rock-star | thunder-stone |
    9. | pikachu-belle | thunder-stone |
    10. | pikachu-pop-star | thunder-stone |
    11. | pikachu-phd | thunder-stone |
    12. | pikachu-libre | thunder-stone |
    13. | pikachu-cosplay | thunder-stone |
    14. +-------------------+----------------------+
    15. (The extra Pikachu are special "cosplay" Pikachu. They're very cute.)
26. For a given pokemon, list its evolution chain / all its evolved / pre-evolved forms.
    1. Typing in ‘eevee’ gives you
    2. +--------------+
    3. | Pokemon |
    4. +--------------+
    5. | eevee |
    6. | vaporeon |
    7. | jolteon |
    8. | flareon |
    9. | espeon |
    10. | umbreon |
    11. | leafeon |
    12. | glaceon |
    13. | sylveon |
    14. +------------+

**8. Full Relational Table specification of DB in the SQL DDL**

CREATE TABLE abilities (

id INT NOT NULL,

identifier VARCHAR(14) CHARACTER SET utf8 NOT NULL,

generation\_id INT NOT NULL,

is\_main\_series INT NOT NULL,

PRIMARY KEY(id)

);

CREATE TABLE encounters (

id INT NOT NULL,

location\_area\_id INT NOT NULL,

encounter\_slot\_id INT NOT NULL,

pokemon\_id INT NOT NULL,

min\_level INT NOT NULL,

max\_level INT NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE evolution\_triggers (

id INT NOT NULL,

identifier VARCHAR(8) CHARACTER SET utf8 NOT NULL,

PRIMARY KEY ( id)

);

CREATE TABLE genders (

id INT NOT NULL,

identifier VARCHAR(10) CHARACTER SET utf8 NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE generations (

id INT NOT NULL,

main\_region\_id INT NOT NULL,

identifier VARCHAR(14) CHARACTER SET utf8 NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE items (

id INT NOT NULL,

identifier VARCHAR(16) CHARACTER SET utf8 NOT NULL,

category\_id INT NOT NULL,

cost INT NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE locations (

id INT NOT NULL,

region\_id INT,

identifier VARCHAR(23) CHARACTER SET utf8 NOT NULL,

PRIMARY KEY(id)

);

CREATE TABLE moves (

id INT NOT NULL,

identifier VARCHAR(16) CHARACTER SET utf8 NOT NULL,

generation\_id INT NOT NULL,

type\_id INT NOT NULL,

strength INT,

pp INT,

accuracy INT,

priority INT NOT NULL,

target\_id INT NOT NULL,

damage\_class\_id INT NOT NULL,

effect\_id INT NOT NULL,

effect\_chance INT,

PRIMARY KEY (id)

);

CREATE TABLE pokemon (

id INT NOT NULL,

identifier VARCHAR(21) CHARACTER SET utf8 NOT NULL,

species\_id INT NOT NULL,

height INT NOT NULL,

weight INT NOT NULL,

base\_experience INT NOT NULL,

is\_default INT NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE pokemon\_abilities (

pokemon\_id INT NOT NULL,

ability\_id INT NOT NULL,

is\_hidden INT NOT NULL,

PRIMARY KEY (pokemon\_id)

);

CREATE TABLE pokemon\_evolution (

id INT NOT NULL,

evolved\_species\_id INT NOT NULL,

evolution\_trigger\_id INT NOT NULL,

trigger\_item\_id INT,

minimum\_level INT,

gender\_id INT,

location\_id INT,

held\_item\_id INT,

time\_of\_day VARCHAR(5) CHARACTER SET utf8,

known\_move\_id INT,

known\_move\_type\_id INT,

minimum\_happiness INT,

minimum\_beauty INT,

minimum\_affection INT,

relative\_physical\_stats INT,

party\_species\_id INT,

party\_type\_id INT,

trade\_species\_id INT,

needs\_overworld\_rain INT,

turn\_upside\_down INT,

PRIMARY KEY (id)

);

CREATE TABLE pokemon\_form\_generations (

pokemon\_form\_id INT NOT NULL,

generation\_id INT NOT NULL,

PRIMARY KEY (pokemon\_form\_id)

);

CREATE TABLE pokemon\_moves (

pokemon\_id INT NOT NULL,

move\_id INT NOT NULL,

level INT NOT NULL

);

CREATE TABLE pokemon\_species (

id INT NOT NULL,

identifier VARCHAR(11) CHARACTER SET utf8 NOT NULL,

generation\_id INT NOT NULL,

evolves\_from\_species\_id INT,

evolution\_chain\_id INT,

color\_id INT NOT NULL,

shape\_id INT NOT NULL,

habitat\_id INT ,

gender\_rate INT NOT NULL,

capture\_rate INT NOT NULL,

base\_happiness INT NOT NULL,

is\_baby INT NOT NULL,

hatch\_counter INT NOT NULL,

has\_gender\_differences INT NOT NULL,

growth\_rate\_id INT NOT NULL

);

CREATE TABLE pokemon\_stats (

pokemon\_id INT NOT NULL,

stat\_id INT NOT NULL,

base\_stat INT NOT NULL,

effort INT NOT NULL

);

CREATE TABLE pokemon\_types (

pokemon\_id INT NOT NULL,

type\_id INT NOT NULL,

slot INT NOT NULL

);

CREATE TABLE regions (

id INT NOT NULL,

identifier VARCHAR(6) CHARACTER SET utf8 NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE types (

id INT NOT NULL,

identifier VARCHAR(8) CHARACTER SET utf8 NOT NULL,

generation\_id INT NOT NULL,

damage\_class\_id INT,

PRIMARY KEY (id)

);

**9. Hard copy of all SQL code w/ comments, as well as any other data acquisition and input programs (processing).**

Included in the Github Repo are

* two csv-processing java files
* sql commands populating the database
  + Located in sql\_commands folder off of the main directory.
* Stored Procedures for each potential question to ask the Database
  + Located in the main directory as file: sql\_stored\_procedures.sql