Who's That Pokemon?

Pokemon Database Final Report

Jundong Hu, Yezheng Li, Kaiwen Wei, and Ruijie Cao[1]

University of Pennsylvania
e-mail: caor@seas.upenn.edu, jundongh@seas.upenn.edu, yezheng@seas.upenn.edu, wkaiwen@seas.upenn.edu

Written on Dec 09, 2019

ABSTRACT

Aims. To construct a Pokemon Database for Pokemon players to search for any Pokemon appeared in all generations, our team has established a Database using Amazon Website Services (AWS) to accomplish this task.
Methods. We collected data from Kaggle.com and uploaded to AWS. With OracleDB, we are able to retrieve data from AWS using our complex optimized queries and displayed the results on our database. Our Pokemon database is also publicly accessible.
Results. The database is fully functional with a signup and login page and the users can search Pokemon by names or Pokedex or types to find the useful information. In addiiton, Pokemon Go popular cities and Pokemon Movies are also suggested for the users.

Key words. Pokemon, Database, Pikachu

1. Introduction

Pokemon is a series of video games developed by Game Freak and published by Nintendo. In 2016, Pokemon Go was launched and becomes a popular game worldwide with a 147 million active users in May 2019. In addition, the Pokemon anime has been popular since their first episode in April 1997. Millions of fans not only love the game but also watch the anime of the every single new generation of Pokemon. There are seven generations in total and more than 800 Pokemons and 18 types of Pokemon, including fire, water, ice and so on. Our goal of this project is to implement complex sql queries and design an optimized Pokemon database. In other words, the project is to develop a user-friendly database platform for Pokemon game players, Pokemon Anime fans, and Pokemon Go players to access the information of each Pokemon and also their opponents.
Our database helps users look up a Pokemon by typing in its name or inputing Pokedex of the Pokemon and the basic information will be printed on the screen such as Pokedex number, name, height, weight, abilities and a picture of the Pokemon. Each Pokemon has a unique Pokedex number and a name, the new Pokemon players may not be familiar with all of them. Even the experienced Pokemon players may not be able to memorize all the information of each Pokemon. In addition, each Pokemon has various different sets of abilities and the abilities can apply different amounts of damage on different types of Pokemon.
Moreover, users can choose a specific type they want to search for and the database will output the top 10 best Pokemon of each type as well as the abilities against the type. The spawn rate of this type in urban area will also be printed for Pokemon Go players.
We also have some features for Pokemon anime fans. The page "movies" contain all the Pokemon movie posters and a link to the trailers. Therefore, the the fans can just look at which one they have not watched and just watch the trailers to decide if they would like to purchase the movies on Netflix or Amazon prime or YouTube.
The last page is shown as the top 3 Pokemon Go popular cities, the real-time temperature and weather are shown in the column as well. The users can search any cities in the search bar to find out the weather. And a google link is provided for each popular city, so if the fans would like to travel to the city, they can find a way and make a plan of how to get there.
Overall, we have a security feature which is login and signup page for the users. Only the users can signed up or logged in can view our database. Otherwise, the users can only see the home page, which is the only introduction and overview of the database. The database is publicly and easily accessible.
Therefore, our database can be used as a game guide for Pokemon Go and Pokemon Anime fans to get the information and increase their win rate of their battles with other players.
Our team members are Jundong Hu, Kaiwe Wei, Yezheng Li, and Ruijie Cao. Jundong has done SQL queries, webpage design, SQL optimization, database connection setup, final report, final presentation, data cleaning, ER diagram. Kaiwen has done database connection setup, SQL queries, webpage design, final report, final presentation, data cleaning, ER diagram. Yezheng has done database connection setup, SQL queries, AWS setup, final report, final presentation, data cleaning, NoSQL and Neo4J testing. And Ruijie has done database connection setup, SQL queries, webpage design, final report, fi-

nal presentation, data cleaning. We have done most of the work together and helped each other.

2. Architecture

2.1. Architecture Overview

1. We stored all our data in AWS/RDS OracleDB and used SQL to extract query results from database POKEMON;
2. HTML is definitely for website design. We used HTML to demonstrate Pokemon images, Pokemon movies, links to Google maps as well as tables of Pokemon basic information.
3. NodeJS (JavaScript) operates the same role as the one in homework 2. We built our database based on the homework 2 architecture.

2.2. Architecture Details

Speaking of organization of web pages,

1. **Welcome Page**:
   "Welcome Page" consists of "Login Section", "Pokemon Hub Overview", "Our Team", "Contact Us" as well as a a sidebar (to view all pages).
   - "Login Section" you can input "Username" and "Password" and submit. Sidebar can only be fully viewed after you login.
   - If you are not a member of our database, you can click sign up to sign up an account.
   - "Pokemon Hub Overview", "Our Team" are just introduction of the project.
   - "Contact Us" section provides the place you can leave messages (as well as your email).
2. **Search-Pokemon Page**:
   "Search-Pokemon Page" is CIS550POKEMON (SEARCH) the page we can input either Pokemon name or Pokedex number to obtain information. For the following example, we used "bulbasaur" and its Pokedex "1" as an example. We presented to queries, one is before optimization and the other one is after to compare the performance. The detailed information of how we optimized the queries was explained in the following section.
   We also show an image of corresponding Pokemon on the page.
3. **Search-by-Type Page**:
   On "Search-by-Type Page", one selects TYPE from dropdown menu and obtains three following results:
   (a) "Spawn Rate in the urban area" for that TYPE;
   (b) "Top 10 Best Pokemon" for that TYPE. We sorted using attack base stat and special attack base stat in descending order, if two Pokemons are tied with the attack value, we sort them using defense base stat and special defense base stat in descending order. If the defense stat is tied again, we use speed to sort in descending order. Similarly the two queries before and after optimization are listed below. CIS550POKEMON (TYPE).
   (c) "Abilities against it" not only shows the abilities that is resistant by the type but also shows what abilities that the type is vulnerable to. CIS550POKEMON (TYPE).
4. **Movies Page**:
   This page provides links to YouTube Movies of corresponding Pokemon movies.
5. **Pokemon-Go-Nest Page**:
   This page demonstrates the top three cities that Pokemon appear. By the way, it provides links to corresponding Google Map locations to help users locate and plan a trip to the cities.

3. Data

1. Pokemon names/pictures: This includes the Pokemon names, their primary and secondary types (if applicable), and also their images.
   `https://www.kaggle.com/vishalsubbiah/pokemon-images-and-types#pokemon.csv` (16 KB, 810 rows, 3 attributes(Name, Type1, Type2))
   Pokemon images are used on CIS550POKEMON (SEARCH).
2. Pokemon stats: This is the main table we will be using to get Pokemon's stats.
   `https://www.kaggle.com/mrdew25/pokemon-database` (394 KB, 928 rows, 42 attributes)
   Together with Pokemon names, this is used in second table "Top 10 Best Pokemon of" of CIS550POKEMON (TYPE).
3. Pokemon movesets: This is the table we can look up Pokemon's moves and can be used for sql query implementation.
   `https://www.kaggle.com/rounakbanik/pokemon` (161 KB, 802 rows, 34 attributes(Name, Type1, Type2))
   It is used in "Abilities against it" in CIS550POKEMON (TYPE).
4. Pokemon Go:
   `https://www.kaggle.com/semioniy/predictemall#300k.csv` (763 MB, 296,021 rows, 208 attributes)
   It is used to derive "Spawn Rate In Urban" in CIS550POKEMON (SEARCH) as well as "Spawn Rate in the urban area" in CIS550POKEMON (TYPE).
5. Pokemon Images:
   `https://www.kaggle.com/kvpratama/pokemon-images-dataset` (39 MB, 809 images)
   This dataset only has better quality of images for all the pokemons.

4. Database Explanation

The credentials required to access our AWS OracleDB database are listed as the following:

Port	1521
Database Name	pokemon
Username	admin
Password	cis550pokemon

Table 1. Credentials to Connect AWS OracleDB

The host name/endpoint is cis550pokemon.ca4mce6zf03f.us-east-1.rds.amazonaws.com.

Because we already deployed our database to AWS, to directly access our database, please use the following link: `ec2-18-223-16-110.us-east-2.compute.amazonaws.com:8082/`

1. Pokemon names/pictures:
   This dataset includes the Pokemon names, their primary and secondary types, and images of them. We do not have to drop any rows because all the Pokemons are not identical.

   (Name, type1, type2)

2. Pokemon abilities:
   This dataset is the main table we will be using to get Pokemon's stats. For this dataset, we are going to drop the duplicates and drop the columns which are already in other tables and keep the ones unique into our database as SQL DDL statements shown above.

   (Name, Abilities, Against_Bug, Against_Dark, Against_Dragon, Against_Electric, Against_Fairy, Against_Fight, Against_Fire, Against_Flying, Against_Ghost, Against_Grass, Against_Ground, Against_Ice, Against_Normal, Against_Poison, Against_Psychic, Against_Rock, Against_Steel, Against_Water)

3. Pokemon EV:
   This dataset is the table we can look up Pokemon's base stat values and can be used for sql query implementation. In this dataset, the pokemon number and name are primary keys and they are used as common parameters to join with other tables. There are 809 Pokemons are included in this table, the most important variable is the type of pokemon, there are either 1 or 2 types of each Pokemon and they can be distributed by the types in 18 groups. The attack, health, height and weight statistics are kept and other unimportant values such as a bunch of EV are dropped.

   (Pokedexnumber, Pokemonname, Height, Weight, Primarytype, Secondarytype, Healthbasestat, Attacbasestat, Defensebasestat, SpecialAttackBaseStat, SpecialDefenseBaseStat, SpeedBaseStat)

4. Pokemon Go:
   This dataset is the table we can look up the top cities that Pokemons appear and can be used for sql query implementation. In this dataset, the area, cities, appearing time of each Pokemon and the environment data of the place are involved. The Pokemon ID is the primary key in this table and also the parameter to join with other tables. This dataset has more than 290,000 Pokemons' sightings. The columns such as cellID, COOC are dropped as they are dirty and unimportant.
   (PokemonID, latitude, longitude, appearedlocaltime, appearedtimeofday, appearhour, appearminute, appeareddayofweek, appeareday, appearedmonth, appearedyear, terrantype, closetowater, city, continent, weather, temperature, windspeed, winbearing, pressure, weathericon, populationdensity, urban, suburban, midurban, rural)

5. Pokemon Images:
   This dataset is the one we can capture the images of overall 809 Pokemon. All of them have unique Pokemon ID and this parameter can be used in the searching performance.

5. Queries

5.1. Sample Query 1

"What are the basic stats of a specific Pokemon and how does it look like? "
We can input either Pokemon name or Pokedex number to obtain information. For the following example, we used "bulbasaur" and its Pokedex "1" as an example. We presented to queries, one is before optimization and the other one is after to compare the performance. The detailed information of how we optimized the queries was explained in the following section.
Before Query Optimization: 0.013s

```
SELECT p.Name, p.type1, p.type2, a.abilities,
e.height, e.weight
FROM PokemonNameType p, Abilities a, EV e
WHERE LOWER(p.Name) = 'bulbasaur'
AND LOWER(a.Name) = 'bulbasaur'
AND LOWER(e.pokemonname) = 'bulbasaur';
```

After Query Optimization: 0.011s

```
SELECT b.POKEDEXNUMBER, b.pokemonname, p.type1,
p.type2, a.abilities, b.height, b.weight
FROM (
SELECT *
FROM EV e
WHERE LOWER(e.pokemonname) = LOWER('bulbasaur')
OR e.pokedexnumber = 1) b
JOIN ABILITIES a
ON LOWER(b.pokemonname) = LOWER(a.Name)
JOIN pokemonnametype p
ON LOWER(a.Name) = LOWER(p.name);
```

5.2. Sample Query 2

"What are the top 10 powerful (sum of highest attack and special attack) Pokemon with type of fire?"
This appears as second table "Top 10 Best Pokemon of" on page. We sorted using attack base stat and special attack base stat in descending order, if two Pokemons are tied with the attack value, we sort them using defense base stat and special defense base stat in descending order. If the defense stat is tied again, we use speed to sort in descending order. Similarly the two queries before and after optimization are listed below.
CIS550POKEMON (TYPE).
Before Query Optimization: 0.016s

```
SELECT *
FROM (SELECT p.Name, e.AttackBaseStat,
e.SpecialAttackBaseStat
FROM EV e, PokemonNameType p
WHERE p.type1 = 'Fire'
```

```
AND LOWER(e.PokemonName) = LOWER(p.Name)
ORDER BY AttackBaseStat DESC,
SpecialAttackBaseStat DESC) a
WHERE ROWNUM<=10;
```

After Query Optimization: 0.014s

```
SELECT *
FROM (SELECT p.Name, e.AttackBaseStat,
e.SpecialAttackBaseStat
FROM EV e
JOIN PokemonNameType p
ON LOWER(e.PokemonName) = LOWER(p.Name)
WHERE p.type1 = 'Fire'
ORDER BY AttackBaseStat DESC,
SpecialAttackBaseStat DESC) a
WHERE ROWNUM<=10;
```

5.3. Sample Query 3

"What are the best abilities to battle against a certain type (ice)?"
This is shown as "Abilities against it" on page CIS550POKEMON (SEARCH). It not only shows the abilities that is resistant by the type and it also shows what abilities that the type is vulnerable to. CIS550POKEMON (TYPE).
Before Query Optimization: 0.019s

```
WITH temp1 AS (SELECT abilities,
1/against AS against,
ROWNUM AS row_a
FROM (SELECT DISTINCT a.abilities,
    a.Against_ice AS against
    FROM Abilities a, EV e
    WHERE a.Against_ice <= 1
    AND a.Against_ice > 0
    AND a.name = e.PokemonName
    ORDER BY Against_ice ASC)
    WHERE ROWNUM <= 10
),
temp2 AS (SELECT abilities,
1/against AS against,
ROWNUM AS row_b
FROM (SELECT DISTINCT a.abilities,
    a.Against_ice AS against
    FROM Abilities a, EV e
    WHERE a.Against_ice >= 2
    AND a.name = e.PokemonName
    ORDER BY Against_ice DESC)
    WHERE ROWNUM <= 10
)
SELECT t1.abilities AS a1,
ROUND(t1.against,2) AS a2, t2.abilities AS a3,
t2.against AS a4
FROM temp1 t1, temp2 t2
WHERE t1.row_a = t2.row_b;
```

After Query Optimization: 0.013s

```
WITH temp1 AS (SELECT abilities,
1/against AS against,
ROWNUM AS row_a
FROM (SELECT DISTINCT a.abilities,
    a.Against_ice AS against
    FROM Abilities a
    JOIN EV e ON a.name = e.PokemonName
    WHERE a.Against_ice <= 1
    AND a.Against_ice > 0
    ORDER BY Against_ice ASC)
    WHERE ROWNUM <= 10
),
temp2 AS (SELECT abilities,
1/against AS against,
ROWNUM AS row_b
FROM (SELECT DISTINCT a.abilities,
    a.Against_ice AS against
    FROM Abilities a
    JOIN EV e ON a.name = e.PokemonName
    WHERE a.Against_ice >= 2
    ORDER BY Against_ice DESC)
    WHERE ROWNUM <= 10
)
SELECT t1.abilities AS a1,
ROUND(t1.against,2) AS a2, t2.abilities AS a3,
t2.against AS a4
FROM temp1 t1
full JOIN temp2 t2
ON t1.row_a = t2.row_b;
```

5.4. Sample Query 4

"What are the top 3 cities that Pokemon spawned in mostly?"
This is on page CIS550POKEMON (NEST) and we obtained three cities "New York", "Chicago", "Prague" and showed them in Pogemon Go Nest page. Because this query is just a simple groupby and count query, we did not try to optimize it.

```
SELECT *
FROM (SELECT City, Count(*)
FROM PokemonGo
GROUP BY City
ORDER BY Count(*) DESC)
WHERE ROWNUM <= 3;
```

5.5. Sample Query 5

"How often/what is the spawn rate of "Abra" (or type fire) in urban areas?"
For each pokemon, it appears on CIS550POKEMON (SEARCH) and for each type, it appears as "Spawn Rate in the urban area" on CIS550POKEMON (TYPE). It is important to help users stay safe, so we only output the spawn rate of Pokemon in urban area rather than rural area. Be aware of surroundings while playing Pokemon Go. Due to the huge size of the table of Pokemon Go Data, we must optimize this query to increase the speed of retrieving data to give our users a better experience of our database. The two SQL codes below are

calculating the spawn rate of "Abra" in urban areas before and after optimization.
Before Query Optimization: 0.313s

```
WITH Temp1 AS (
  SELECT Count(*) AS num
  FROM PokemonGo pg
  JOIN EV e
  ON pg.pokemonID = e.pokedexnumber
  WHERE pg.urban = 'true' AND
  LOWER(e.pokemonname) = 'abra'
),
Temp2 AS (
  SELECT Count(*) AS num
  FROM PokemonGo pg
  JOIN EV e
  ON pg.pokemonID = e.pokedexnumber
  WHERE LOWER(e.pokemonname) = 'abra'
),
Temp3 AS (SELECT CASE WHEN t2.num <> 0
THEN t1.num/t2.num ELSE NULL END AS SpawnRateUrban
  FROM Temp1 t1, Temp2 t2
)
SELECT e.POKEDEXNUMBER, e.pokemonname,
p.type1, p.type2, a.abilities,
ROUND(e.height, 2) AS height,
ROUND(e.weight, 2) AS weight,
ROUND(t.SpawnRateUrban,2) AS SpawnRateUrban
FROM EV e, temp3 t, ABILITIES a, pokemonnametype p
WHERE LOWER(e.pokemonname) = LOWER('abra')
AND LOWER(e.pokemonname) = LOWER(a.Name)
AND LOWER(a.Name) = LOWER(p.name)
```

After Query Optimization: 0.221s

```
WITH Temp1 AS (
  SELECT Count(*) AS num
  FROM PokemonGo pg
  JOIN EV e
  ON pg.pokemonID = e.pokedexnumber
  WHERE pg.urban = 'true'
  AND LOWER(e.pokemonname) = 'abra'
),
  Temp2 AS (
  SELECT Count(*) AS num
  FROM PokemonGo pg
  JOIN EV e
  ON pg.pokemonID = e.pokedexnumber
  WHERE LOWER(e.pokemonname) = 'abra'
),
  Temp3 AS (SELECT CASE WHEN t2.num <> 0
  THEN t1.num/t2.num
  ELSE NULL END AS SpawnRateUrban
  FROM Temp1 t1, Temp2 t2
)
SELECT b.POKEDEXNUMBER, b.pokemonname,
p.type1, p.type2, a.abilities,
ROUND(b.height, 2) AS height,
ROUND(b.weight, 2) AS weight,
ROUND(b.SpawnRateUrban,2) AS SpawnRateUrban
FROM (
SELECT *
FROM EV e, temp3 t
WHERE LOWER(e.pokemonname) = LOWER('abra')) b
JOIN ABILITIES a
ON LOWER(b.pokemonname) = LOWER(a.Name)
JOIN pokemonnametype p
ON LOWER(a.Name) = LOWER(p.name)
```

The code below is just for demonstration purposes of calculating spawn rate of type fire in urban areas. Due to the similarity of the following code and above, we only showed the one after optimization.

```
SELECT CASE WHEN t2.num <> 0
THEN ROUND(t1.num/t2.num, 3)
ELSE 0
END AS SpawnRateUrban
FROM (
SELECT Count(*) AS num
FROM PokemonGo pg JOIN EV e
ON pg.pokemonID = e.pokedexnumber
WHERE pg.urban = 'true'
AND e.primarytype = 'fire') t1,
(SELECT Count(*) AS num
FROM PokemonGo pg
JOIN EV e ON pg.pokemonID = e.pokedexnumber
WHERE e.primarytype = 'fire') t2
```

6. Performance evaluation

When we were doing the milestone 2, our ER diagram showed a lot of redundancies among the table attributes. As time went by, we learned 2NF and 3NF and BCNF, so we further reduced and picked the attributes to make our data in a form of 3NF and avoid redundancy.
Recorded timings: The timings were provided by SQL developer. We add indices (on all four tables, ABILITIES, EV, POKEMONGO, POKEMONNAMETYPE) to speed up the queries.
We also mentioned query optimization in section by techinques learned from lectures: use JOIN instead of CROSS PRODUCT and push SELECTION and PROJECTION as far as possible, so we decreased the tuples of joins.

7. Technical Challenges

1. The first challenge we had was that we were not able to find a large enough table for Pokemon. There are about 800 Pokemons in total, so most of the tables are only about 800 tuples, but the requirement was at least ten thousand of rows. After more careful searching of database, we finally found a Pokemon Go table which contains sufficient amount of data for our project.
2. At the beginning, we were struggling to have the database connection. We were unable to set the correct PATH while trying to connect our local host to the OracleDB on AWS. Finally after getting help from TAs and peers, we were able to successfully connect to OracleDB.

3. We spent some time trying to display multiple (up to 3) OracleDB JSON results on one HTML page. Initially after extracting every query from OracleDB, we close connection to OracleDB but it seems the result is unstable (i.e. it is likely to failing for some queries). We eventually, close connection to OracleDB after all queries are extracted and the handler turns stable now and we put the JSON results in an array (instead of concatenate multiple JSON objects into one JSON objects) to output the results on HTML page.
4. We managed to overcome difficulties of deploying our server on AWS.
5. We were not able to create a new website/HTML page from the existing page. We did not restart our localhost after we modified the router.get in index.js, so eventually we were able to do it.
6. It took us some time to learn how to code in CSS and HTML. Learning a new language and its syntax took us time, but it was really beneficial. For example, we were not able to change the color of the background or even the round-edge radius. After we watched the tutorials online and read some documents, we were able to construct beautiful websites.

8. Extra credit

1. SQL for the relational data stored with AWS/RDS. We used OracleDB to store data.
2. Using cloud hosting for the application server. Our database website is fully publicly accessible.
   `ec2-18-223-16-110.us-east-2.compute.amazonaws.com:8082/`
3. Implementing security safeguards. We implemented a login and signup page for security purposes.
4. Integrating with other applications to return additional information. We integrated with Google Maps on Pokemon Go Nest page for users to search locations.
5. Adding real-time streaming data. We added real time weather data to our Pokemon Go Nest page for all the popular cities. In addition, the users can search any city in the search bar to find out the weather including temperature and an interesting picture.

Acknowledgements. We would like to give our special thanks to our teaching assistants Rahul Singh Katoch and Eileen Choe, and also our professor Susan Davidson for all the help they provided us to accomplish this project. We also would like to mention W3 school and some online learning resources that we used for developing our database.

References

Search-by-Type Page, (8082/bestof), http://ec2-18-223-16-110.us-east-2.compute.amazonaws.com:8082/bestof
Search-Pokemon Page, (8082/recommendations), http://ec2-18-223-16-110.us-east-2.compute.amazonaws.com:8082/recommendations .
Pokemon-Go-Nest Page, (8082/nest) http://ec2-18-223-16-110.us-east-2.compute.amazonaws.com:8082/nest
Movies Page, (8082/poster), http://ec2-18-223-16-110.us-east-2.compute.amazonaws.com:8082/poster

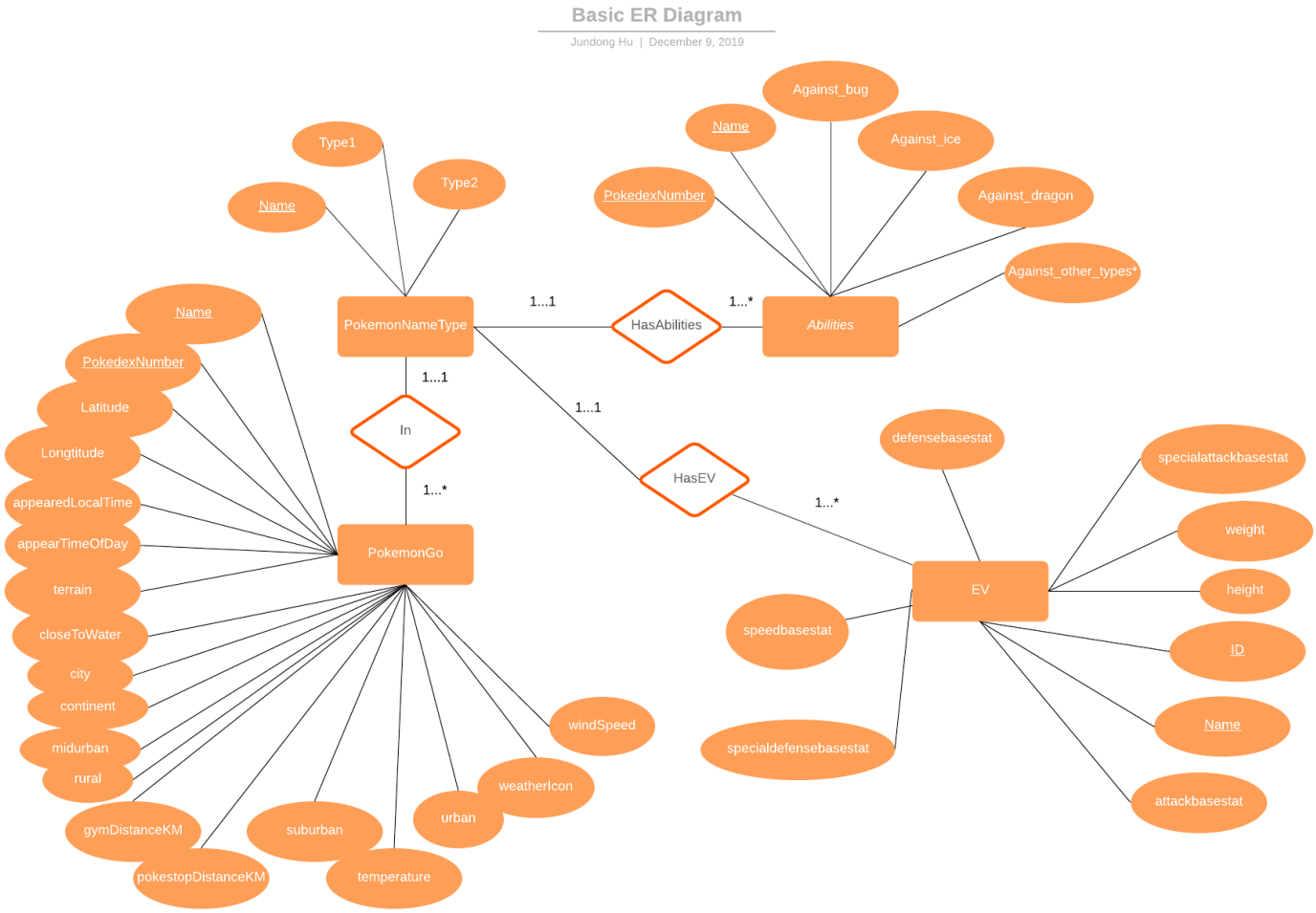

Fig. 1. ER Diagram