DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

PES2201800462 Venkatavaradan R

Pokemon DB is a database that stores the information used in all the Pokemon games. The introduction will introduce you to the transactions of the pokemon miniworld and explain concepts like Pokedex, Types, Evolutions, Trainers, Battles etc.

The schema has 6 major relations. After converting the ERD to table format, we get a total of 11 relations. The dependency diagram shows that the Pokemon and Type table are core parts of this database.

The FD and normalization portion shows us that the relations created by the ERD format are already normalized to a certain extent. We discuss these normalization levels in this segment.

DDL has the scripts for creating the relations. There are 11 relations scripts as mentioned above + 1 more for audit logs, leaving us at a grand total of 12 relations. I have included 10 total triggers for this database. 1 is to automatically insert some data into a table when data is inserted into a certain table, the other 9 are for audit logs on the 3 major tables (one each table for inset, update and delete)

There are 7 queries given of medium-high level including nested queries, outer join queries, and queries using aggregate functions.

In the end, I feel the pokemonDB is robust with a few errors that can be corrected in the future, data redundancy can still be reduced and can be improved by adding multiple more relations that are of use to the competitive pokemon fanbase.

Introduction

Project Topic: Pokemon DB

Introduction to pokemon

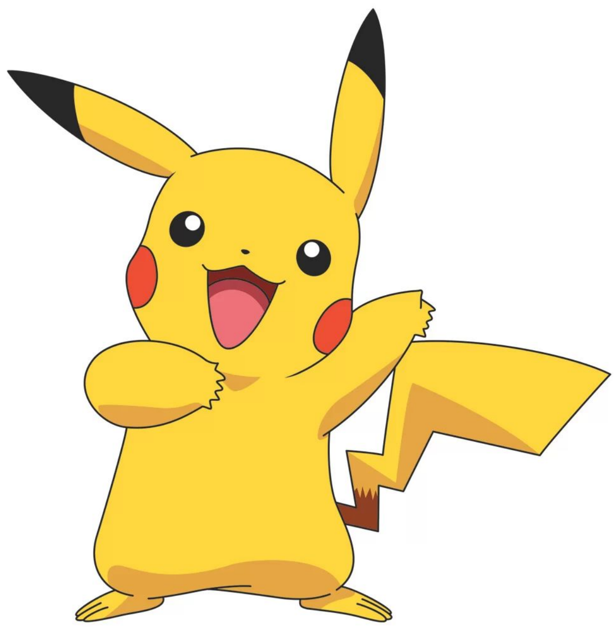
Pokémon, which is short for Pocket Monsters, is about the bond between creatures and the trainers that control them. There are 2 main objectives of the Pokémon games:

1. Collect as many Pokémon as possible
2. Train the caught Pokémon though Pokémon battles

There are various concepts that we can discuss in the pokemon miniworld, but I will be listing out and giving a brief explanation only for the topics relevant to this paper.

Let's start with an example of a pokemon:

Meet **Pikachu**



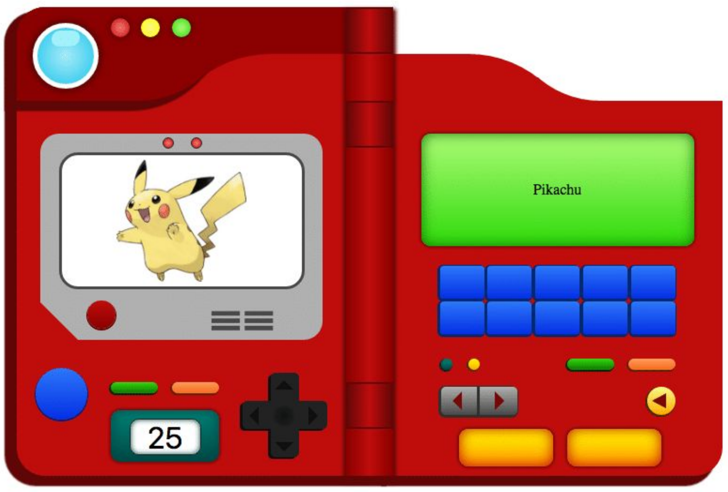
Pikachu

Pikachu is the mascot for pokemon, and we will be using him as an example.

Pokedex and Types

Each Pokemon is a part of a set of **Types**. These types decide what elemental class the pokemon belong to.

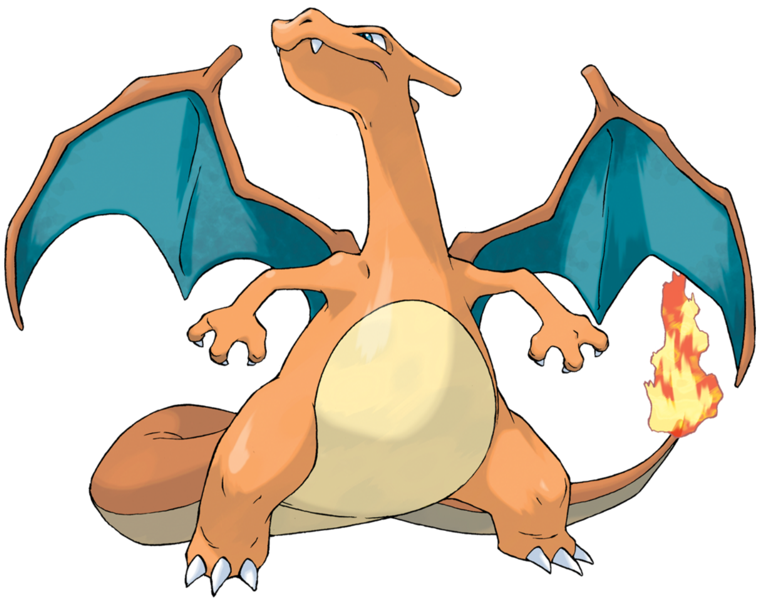
A Pokedex is a device that is used to identify pokemon. Every single pokemon in the Pokemon miniworld has a **Unique ID Number**.



Pokedex

Pikachu is an Electric Type pokemon - ⚡ Electric
Each pokemon can have **1 or 2** types that it belongs to, but not more.

An example of a pokemon with more than 1 type is Charizard



Charizard

Chariard here is a Fire and Flying type pokemon Fire Flying

Each type has its own **weaknesses** and **strengths** against other types when in battle. We will cover pokemon battles and trainers in the following segments.

There are a total of 18 different pokemon types:

BUG
DARK
DRAGON
ELECTRIC
FAIRY
FIGHTING
FIRE
FLYING
GHOST
GRASS
GROUND
ICE
NORMAL
POISON
PSYCHIC
ROCK
STEEL
WATER

Pokemon Types

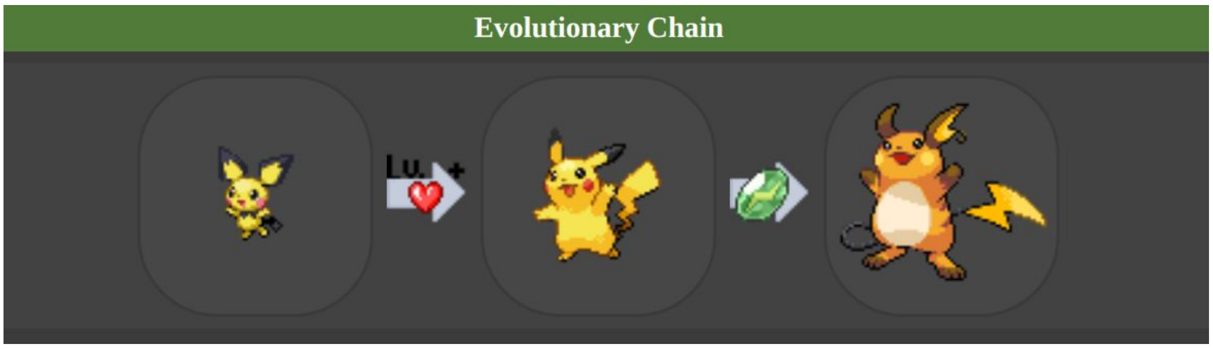
Pokemon Evolutions

Pokemon also have the capability to evolve from their current stage to a more powerful and possibly type-changed stage.

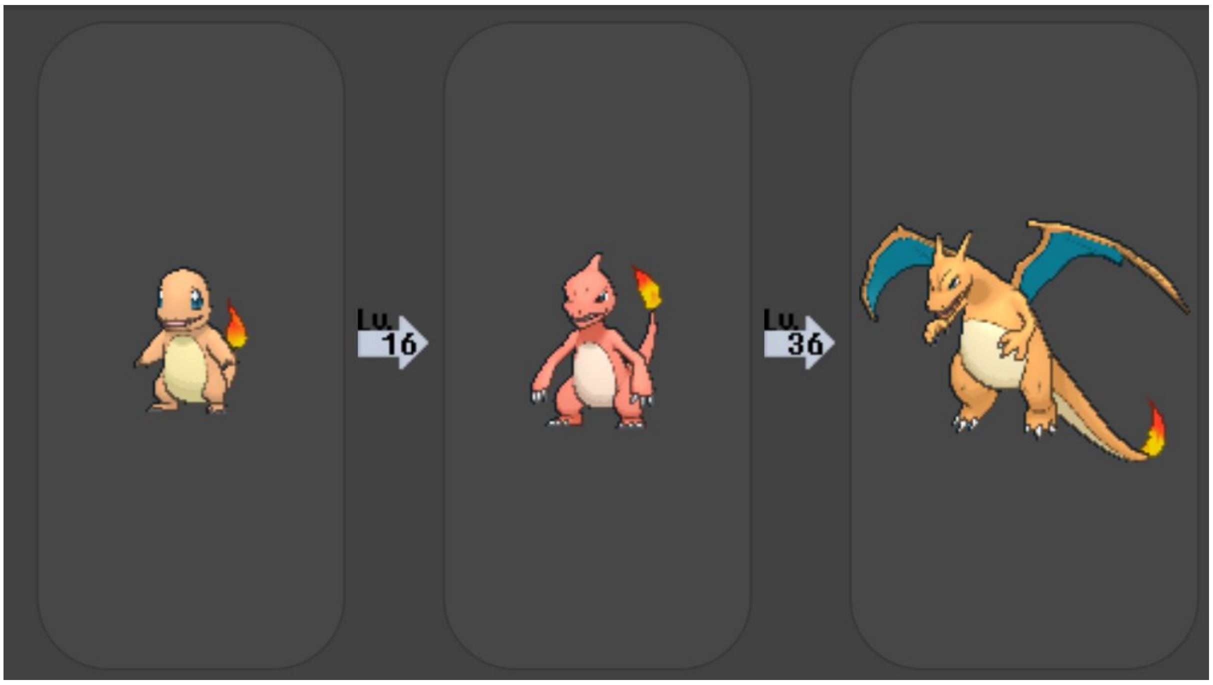
There are a total of 3 different stages:

1. Basic (The lowest stage that a pokemon starts from once it hatches from its egg)
2. 1st Evolution / 1st Stage
3. 2nd Evolution / 2nd Stage

Here are the evolutions for Pikachu (which is a 1st stage pokemon) and Charizard (which is a 2nd stage pokemon)



Evolutionary chain (Pikachu)

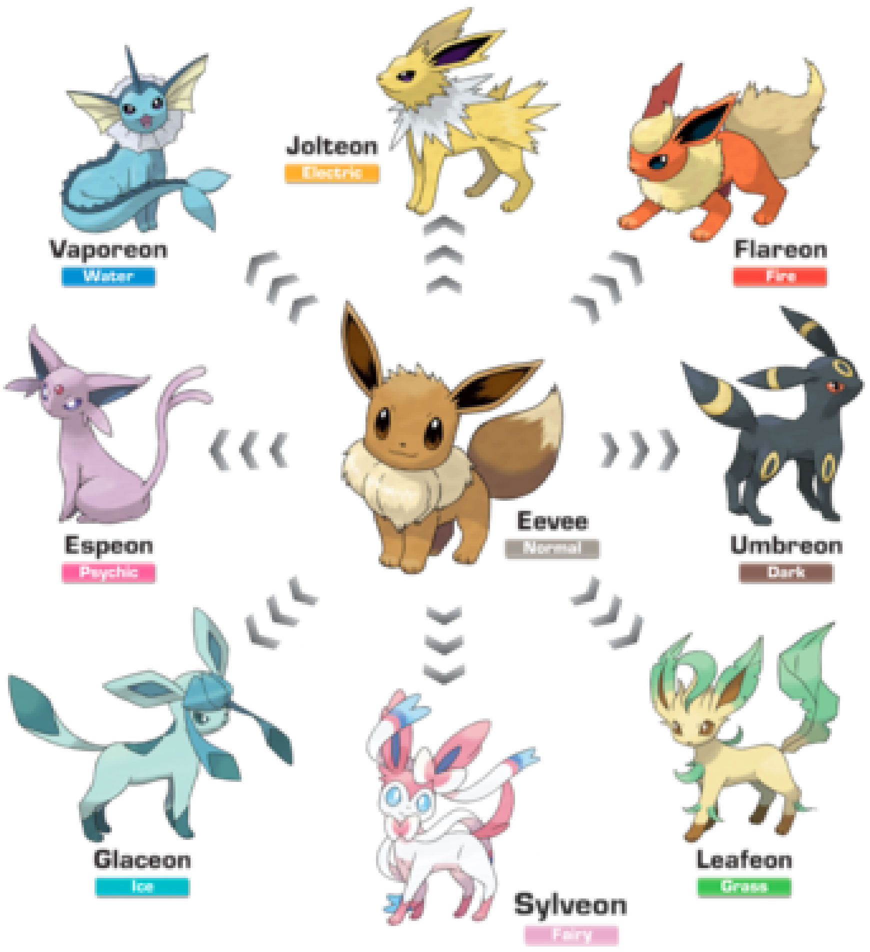


Evolutionary chain (Charizard)

There is no pokemon that can evolve more than twice. However, there are pokemon that can evolve into different pokemon based on what the player chooses. i.e a single pokemon can have multiple different 1st stage evolutions.

The most common example of this is a pokemon named Eevee, and its 8 Evolutions

Eevee



Eeveelutions (Evolutions of Eevee)

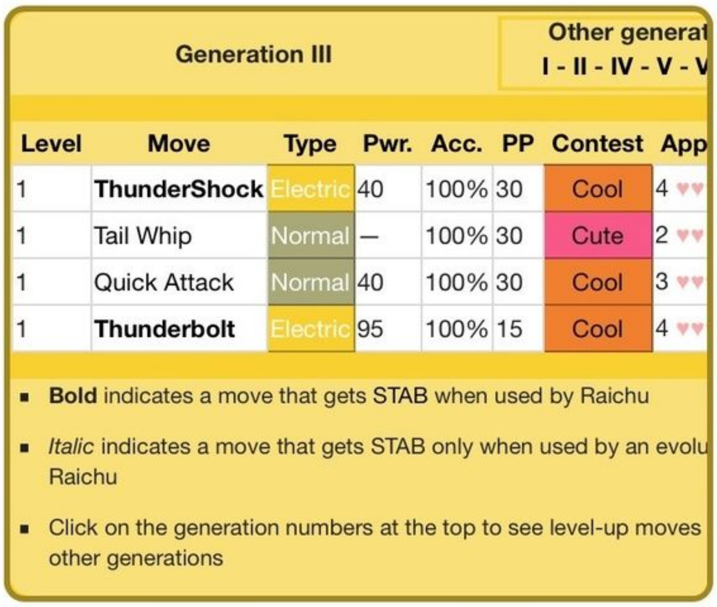
Pokemon Trainers and Pokemon Battles

As mentioned in the beginning of this document, the main objective of the game is to capture all the pokemon and battle against other trainers, but what are pokemon trainers and battles? Let's take a look!

People become pokemon trainers when they get access to a **Trainer License ID**. Trainers are the people who go about capturing pokemon and battling other trainers. Pokemon trainers capture wild pokemon inside pokeballs at whatever location the pokemon appear.

Pokeball

Pokemon Battles occur between 2 pokemon. Each pokemon has a set of **moves** that it uses against the other pokemon, and **each move has only 1 type**.



Generation III						Other generat I - II - IV - V - V	
Level	**Move**	**Type**	**Pwr.**	**Acc.**	**PP**	**Contest**	**App**
1	**ThunderShock**	Electric	40	100%	30	Cool	4 ♥♥
1	Tail Whip	Normal	—	100%	30	Cute	2 ♥♥
1	Quick Attack	Normal	40	100%	30	Cool	3 ♥♥
1	**Thunderbolt**	Electric	95	100%	15	Cool	4 ♥♥

- **Bold** indicates a move that gets STAB when used by Raichu
- *Italic* indicates a move that gets STAB only when used by an evolu Raichu
- Click on the generation numbers at the top to see level-up moves other generations

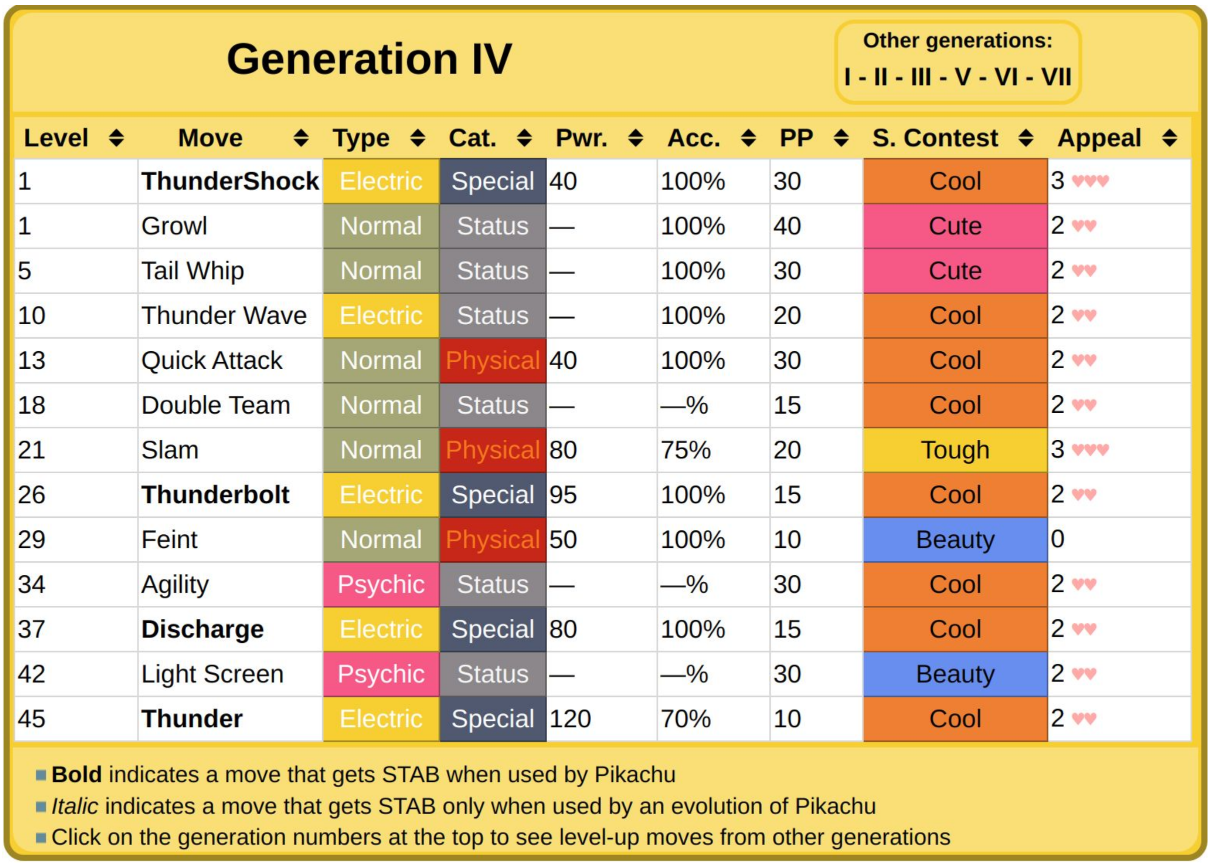
Pokemon Moveset

Here's where type effectiveness comes into play. Let's imagine the 2 pokemon you have been introduced to, **Pikachu** and **Charizard** , are having a pokemon battle against each other.

A smart trainer knows that Electric Electric type moves, such as thunderbolt (see pokemon Moveset picture), are super-effective (stronger) against Flying Flying type pokemon like charizard, and use electric type moves to deal more damage, and hence win the battle.

Another factor that determines whether your pokemon will do good in a pokemon battle is its **level**. The level of a pokemon is basically a measure of its strength. Higher the level, stronger the pokemon.

A pokemon learns **specific moves at a specific levels**. (see Pokemon moveset chart given below)



Generation IV

Other generations:
I - II - III - V - VI - VII

Level	Move	Type	Cat.	Pwr.	Acc.	PP	S. Contest	Appeal
1	**ThunderShock**	Electric	Special	40	100%	30	Cool	3 ♥♥♥
1	Growl	Normal	Status	—	100%	40	Cute	2 ♥♥
5	Tail Whip	Normal	Status	—	100%	30	Cute	2 ♥♥
10	Thunder Wave	Electric	Status	—	100%	20	Cool	2 ♥♥
13	Quick Attack	Normal	Physical	40	100%	30	Cool	2 ♥♥
18	Double Team	Normal	Status	—	—%	15	Cool	2 ♥♥
21	Slam	Normal	Physical	80	75%	20	Tough	3 ♥♥♥
26	**Thunderbolt**	Electric	Special	95	100%	15	Cool	2 ♥♥
29	Feint	Normal	Physical	50	100%	10	Beauty	0
34	Agility	Psychic	Status	—	—%	30	Cool	2 ♥♥
37	**Discharge**	Electric	Special	80	100%	15	Cool	2 ♥♥
42	Light Screen	Psychic	Status	—	—%	30	Beauty	2 ♥♥
45	**Thunder**	Electric	Special	120	70%	10	Cool	2 ♥♥

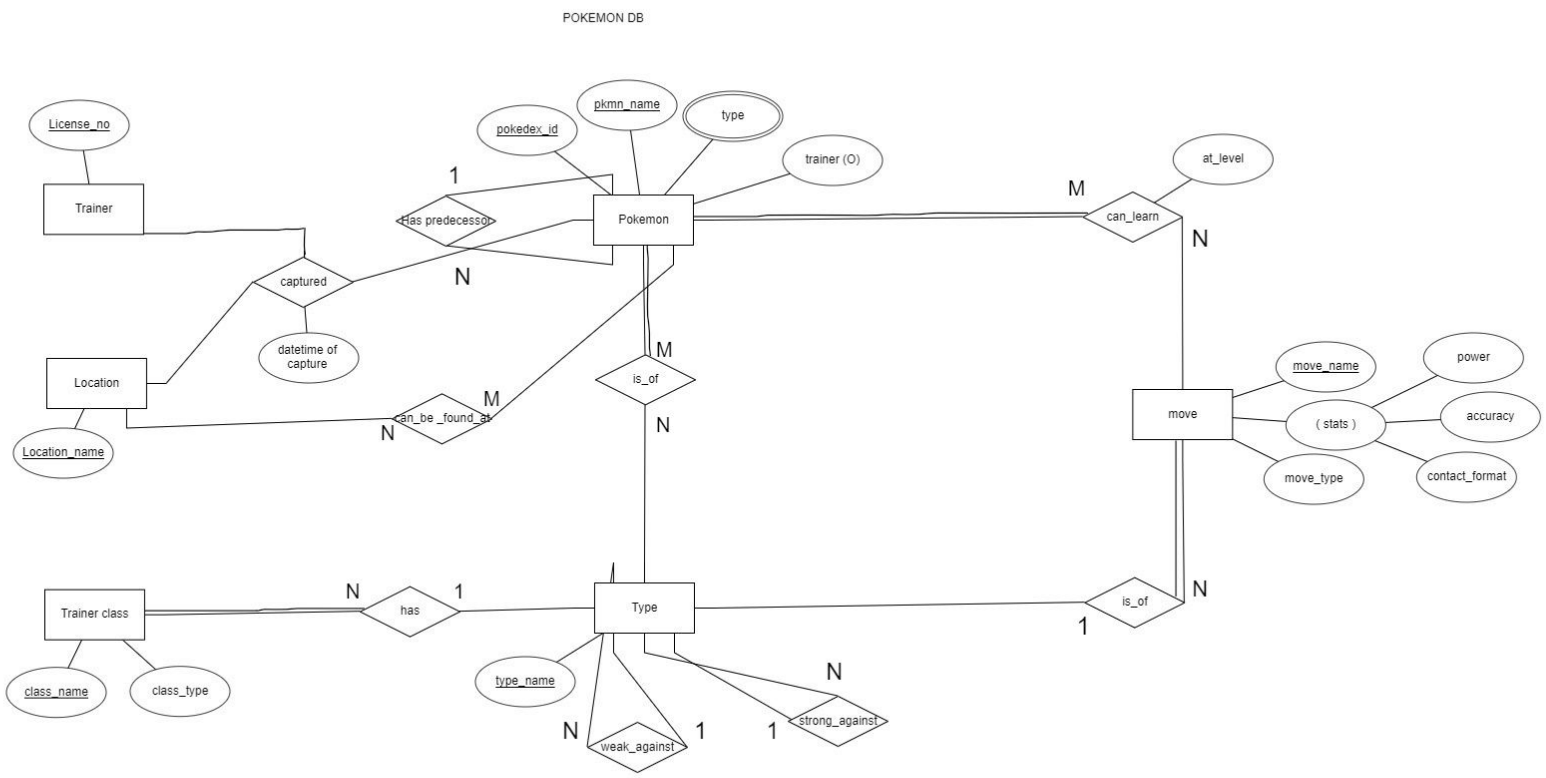
- **Bold** indicates a move that gets STAB when used by Pikachu
- *Italic* indicates a move that gets STAB only when used by an evolution of Pikachu
- Click on the generation numbers at the top to see level-up moves from other generations

Pokemon Moveset (Pikachu)

Ex) Ignoring most of the columns, we can see that Pikachu learns Thunderbolt, an electric Electric type move at the level 26.

Data Model

ERD:



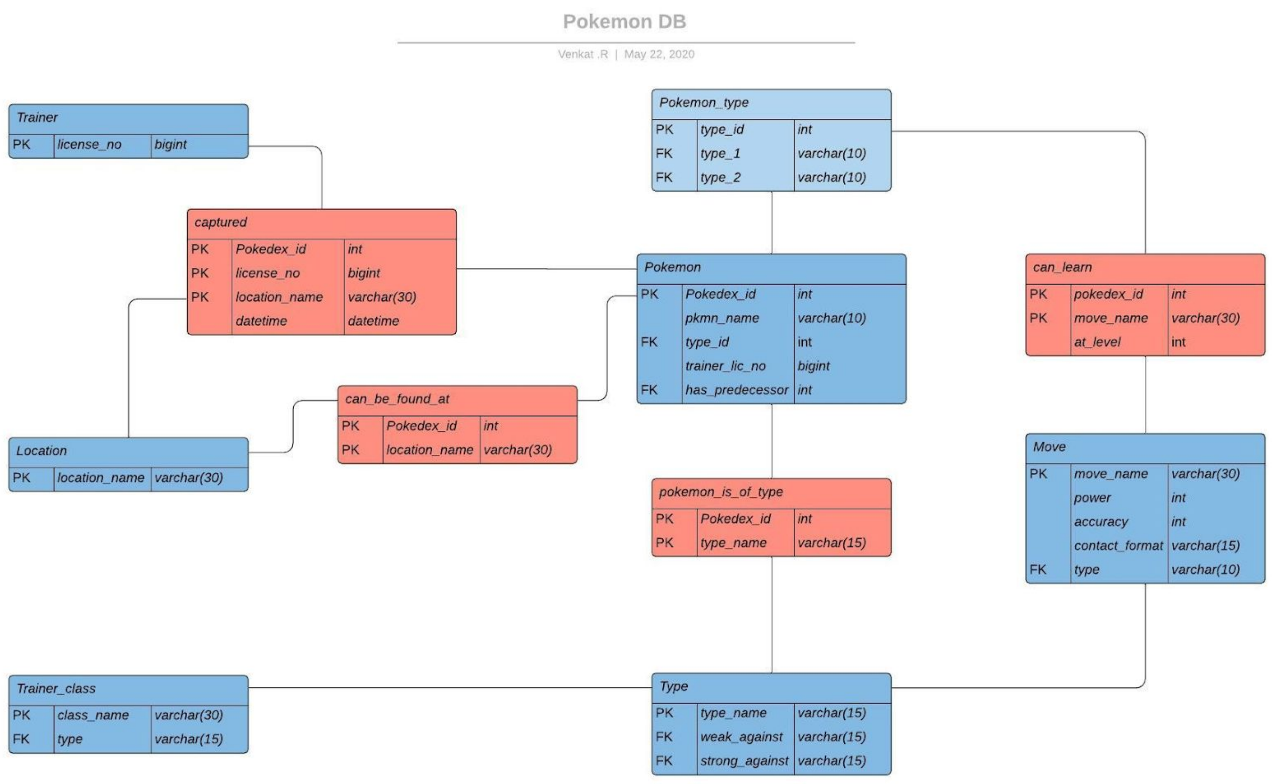
The ERD given above is fairly straightforward. There are 6 entities:

1. **Pokemon**
    a. PK - pokedex_id - As mentioned in the previous section, the pokedex id is unique for every pokemon
2. **Move**
    a. PK - Move_name - There can be no moves with the same name
3. **Type**
    a. PK - Type_name - There can be no 2 types with the same name
4. **Trainer class**
    a. PK - Class_name - There can be no 2 trainer classes that have the same name
5. **Location**
    a. PK - Location_name - There can be no 2 locations that have the same name
6. **Trainer**
    a. PK - License_no - Each trainer has his own license no, similar to an aadhaar card

The relationships are also simple.

1. (Pokemon) **Can_learn** (move)
    a. Any pokemon can learn a move at a specific level.
    b. A pokemon can learn multiple moves and a move can be learned by multiple pokemon
2. (Move) **is_of** (type)
    a. Every move has to have 1 type
    b. There can be many moves of the same type
3. (Type) **has** (trainer class)
    a. Every trainer class has a specific type assigned to them
    b. There can be many trainer classes of the same type
4. (Pokemon) **is_of** (Type)
    a. Every pokemon has to have at least 1 type
    b. Every type can have many pokemon
5. (Pokemon) **can_be_found_at** (Location)
    a. A pokemon can be found at multiple locations, and a location can have multiple pokemon
6. **Captured**
    a. A pokemon is caught by a trainer at a specific location at a specific datetime
7. **Has_predecessor**
    a. A pokemon can have multiple predecessors, but will have only a single successor
        i. A pokemon has 3 stages, but when a pokemon breeds, the egg given always hatches the pokemon in its **basic** stage. i.e it doesnt matter if the parent pokemon is **basic, stage 1 or stage 2**, the successor is always a basic pokemon
8. **Weak_against**
    a. A type can be weak against another type
9. **Strong_against**
    a. A type can be strong against another type

ERD - Conversion to tables(including relationships)



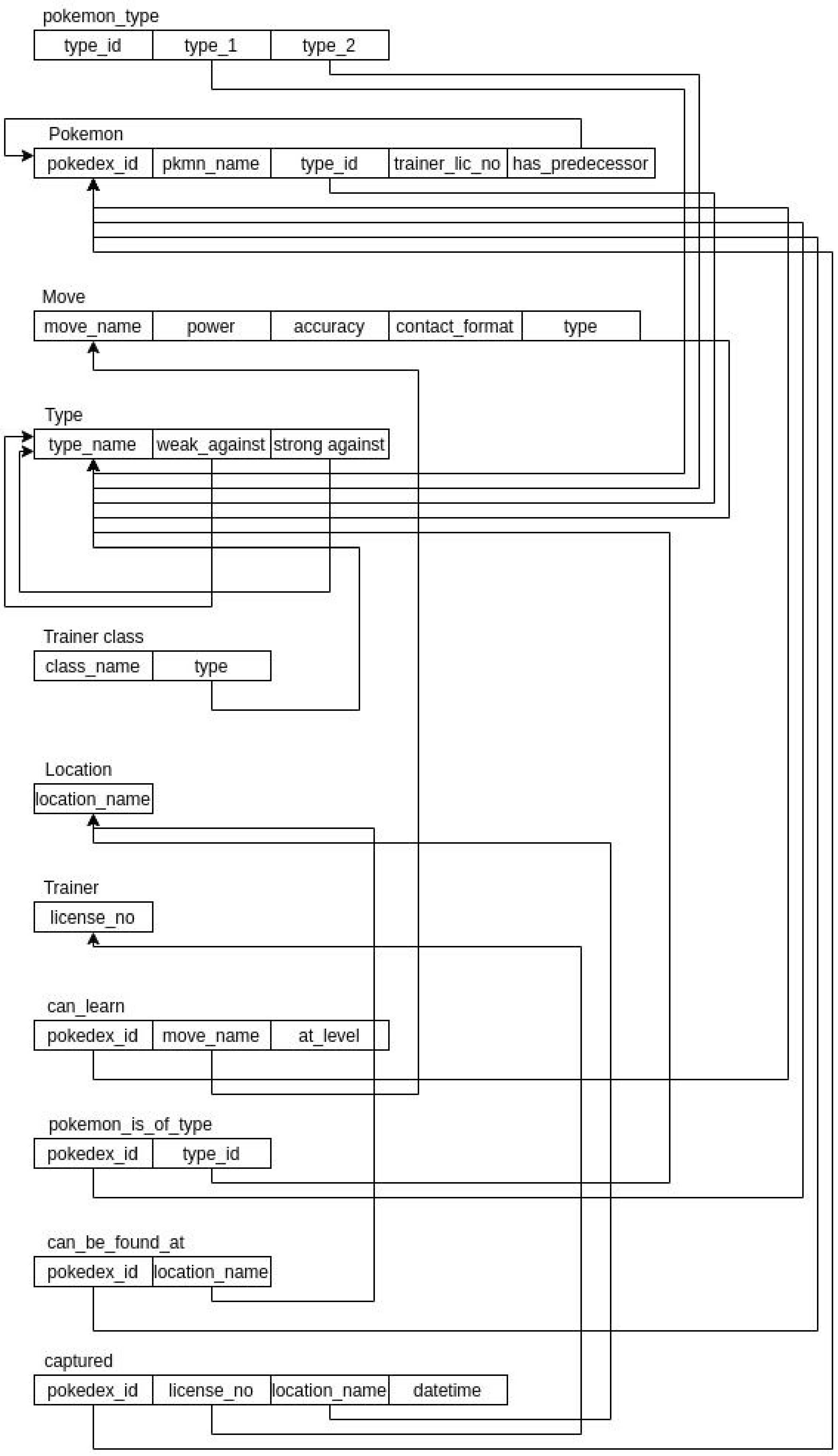
I then converted the first erd into the one shown above, by converting the relationships into tables

Steps:

1. All entities into tables mentioning PK and FK
2. Pokemon_type was multivalued, so i created an extra table with type_id as PK
3. Stats was a composite attribute, so i merged them into the same table as individual elements
4. For all 1:N relationships, I took PK of ‘1’ and added it as FK to ‘N’
5. For all M:N relationships, I created a new table with both PKs
6. To show 1:N recurrence relationships, I added the PK of the as FK to the same table under a different name

Dependency Diagram:

The dependency diagram given below shows the PKs and FKs.
We can see that type_name and pokedex_id are pivotal.
Functional Dependencies are given in the next section



pokemon_type
type_id
type_1
type_2
Pokemon
pokedex_id
pkmn_name
type_id
trainer_lic_no
has_predecessor
Move
move_name
power
accuracy
contact_format
type
Type
type_name
weak_against
strong against
Trainer class
class_name
type
Location
location_name
Trainer
license_no
can_learn
pokedex_id
move_name
at_level
pokemon_is_of_type
pokedex_id
type_id
can_be_found_at
pokedex_id
location_name
captured
pokedex_id
license_no
location_name
datetime

FD and Normalization

Functional dependencies and Normal forms

1NF -> atomic values
2NF -> 1NF + no partial dependencies (There are no cols that depend on only one part of a multi part key)
3NF -> 2NF + no transitional dependencies
4NF -> 3.5NF + no multivalued dependencies
5NF -> 4NF + lossless joins

Note:

1. X -> Z is a transitive dependency if the following three functional dependencies hold true:
   - X->Y
   - Y does not ->X
   - Y->Z
2. **BCNF** or **3.5NF:** for X -> Y, at least one of the following conditions hold true:
   - $X \to Y$ is a trivial functional dependency (Y ⊆ X),
   - *X* is a superkey for schema *R*.

Functional Dependencies: A set of attributes X determines a set of attributes Y if the value of X determines a unique value of Y

- Pokemon_type
  - 3NF
  - Type_id -> {type_1, type_2}
- Pokemon
  - 2NF
  - Pokedex_id -> {pkmn_name, type_id, trainer_lic_no, has_predecessor}
- Move
  - 3NF
  - move_name -> {power, accuracy, contact_format, type}
- Type
  - 3NF
  - type_name -> {weak_against, strong_against}
- Trainer_class
  - 5NF
  - class_name -> {type}
- Location

    - 5NF
    - No FDs
- Trainer
    - 5NF
    - No FDs
- Can_learn
    - 5NF
    - {pokedex_id, move_name} -> at_level
- Pokemon_is_of_type
    - 5NF
    - No FDs
- Can_be_found_at
    - 5NF
    - No FDs
- Captured
    - 5NF
    - {pokedex_id, license_no, location_name} -> datetime

Cases where violations can occur

1NF gets violated if we didn't convert the attribute ‘type’ into its own relation [non atomic]

2NF gets violated if you add pokedex_id to attribute to pokemon_type relation [both types can refer to a single pk, 2 are not necessary]

3NF gets violated for the pokemon relation because it has a candidate key, pokemon_name. If this candidate key was not there, then the relation would be 3NF but it isn't, because there is a transitive dependency present (pokedex_id -> pokemon_name -> type_id)

5NF gets violated if instead of datetime, we use 2 separate attributes date, and time [multivalued dependencies are present]

DDL

Given below are the table scripts.

```
CREATE TABLE type (
   type_name varchar(15) PRIMARY KEY,
   weak_against varchar(15) NOT NULL,
   strong_against varchar(15) NOT NULL,
   CONSTRAINT fk_type_weak_against FOREIGN KEY (weak_against) REFERENCES
type(type_name)
   ON DELETE CASCADE
   ON UPDATE CASCADE,
   CONSTRAINT fk_type_strong_against FOREIGN KEY (strong_against) REFERENCES
type(type_name)
   ON DELETE CASCADE
   ON UPDATE CASCADE
);

CREATE TABLE pokemon_type (
   type_id int PRIMARY KEY,
   type_1 varchar(15),
   type_2 varchar(15),
   CONSTRAINT fk_pokemon_type_type_1 FOREIGN KEY (type_1) REFERENCES
type(type_name)
   ON DELETE SET NULL
   ON UPDATE CASCADE,
   CONSTRAINT fk_pokemon_type_type_2 FOREIGN KEY (type_2) REFERENCES
type(type_name)
   ON DELETE SET NULL
   ON UPDATE CASCADE
);

CREATE TABLE pokemon (
   pokedex_id int PRIMARY KEY,
   pkmn_name varchar(20) NOT NULL,
   type_id int,
   trainer_lic_no bigint NOT NULL,
   has_predecessor int,
   CONSTRAINT fk_pokemon_type_id FOREIGN KEY (type_id) REFERENCES
pokemon_type(type_id)
   ON DELETE SET NULL
   ON UPDATE CASCADE,
   CONSTRAINT fk_pokemon_has_predecessor FOREIGN KEY (has_predecessor)
REFERENCES pokemon(pokedex_id)
   ON DELETE SET NULL
   ON UPDATE CASCADE
);

CREATE TABLE trainer (
   license_no bigint PRIMARY KEY
);

CREATE TABLE location (
   location_name varchar(30) PRIMARY KEY
);

CREATE TABLE trainer_class_ (
   class_name varchar(30) PRIMARY KEY,
   type varchar(15),
   CONSTRAINT fk_trainer_class_type FOREIGN KEY (type) REFERENCES
type(type_name)
   ON DELETE SET NULL
   ON UPDATE CASCADE
);

CREATE TABLE move (
   move_name varchar(30) PRIMARY KEY,
   power int NOT NULL,
   accuracy int NOT NULL,
   contact_format varchar(15) NOT NULL,
   type varchar(15),
   CONSTRAINT fk_move_type FOREIGN KEY (type) REFERENCES type(type_name)
   ON DELETE SET NULL
   ON UPDATE CASCADE
);

CREATE TABLE can_learn (
   pokedex_id int,
   move_name varchar(30),
   at_level int NOT NULL,
   CONSTRAINT pk_can_learn PRIMARY KEY (pokedex_id,move_name),
   CONSTRAINT fk_can_learn_pokedex_id FOREIGN KEY (pokedex_id) REFERENCES
pokemon(pokedex_id)
   ON DELETE CASCADE
   ON UPDATE CASCADE,
   CONSTRAINT fk_can_learn_move_name FOREIGN KEY (move_name) REFERENCES
move(move_name)
   ON DELETE CASCADE
   ON UPDATE CASCADE
);
```

```
CREATE TABLE pokemon_is_of_type (
   pokedex_id int,
   type_name varchar(15),
   CONSTRAINT pk_pokemon_is_of_type PRIMARY KEY (pokedex_id,type_name),
   CONSTRAINT fk_pokemon_is_of_type_pokedex_id FOREIGN KEY (pokedex_id)
REFERENCES pokemon(pokedex_id)
   ON DELETE CASCADE
   ON UPDATE CASCADE,
   CONSTRAINT fk_pokemon_is_of_type_type_name FOREIGN KEY (type_name)
REFERENCES type(type_name)
   ON DELETE CASCADE
   ON UPDATE CASCADE
);

CREATE TABLE can_be_found_at (
   pokedex_id int NOT NULL,
   location_name varchar(30) NOT NULL,
   CONSTRAINT pk_can_be_found_at PRIMARY KEY (pokedex_id,location_name),
   CONSTRAINT fk_can_be_found_at_pokedex_id FOREIGN KEY (pokedex_id)
REFERENCES pokemon(pokedex_id)
   ON DELETE CASCADE
   ON UPDATE CASCADE,
   CONSTRAINT fk_can_be_found_at_location_name FOREIGN KEY (location_name)
REFERENCES location(location_name)
   ON DELETE CASCADE
   ON UPDATE CASCADE
);

CREATE TABLE captured (
   pokedex_id int,
   license_no bigint,
   location_name varchar(30),
   CONSTRAINT pk_captured PRIMARY KEY (pokedex_id,license_no,location_name),
   CONSTRAINT fk_captured_pokedex_id FOREIGN KEY (pokedex_id) REFERENCES
pokemon(pokedex_id)
   ON DELETE CASCADE
   ON UPDATE CASCADE,
   CONSTRAINT fk_captured_license_no FOREIGN KEY (license_no) REFERENCES
trainer(license_no)
   ON DELETE CASCADE
   ON UPDATE CASCADE,
   CONSTRAINT fk_captured_location_name FOREIGN KEY (location_name)
REFERENCES location(location_name)
   ON DELETE CASCADE
   ON UPDATE CASCADE
);
```

```
CREATE TABLE audit_logs(
    id int NOT NULL AUTO_INCREMENT,
    table_altered VARCHAR(255) NOT NULL,
    operation VARCHAR(255) NOT NULL,
    by_user VARCHAR(255) NOT NULL,
    at_time DATETIME NOT NULL,
    PRIMARY KEY (id)
);
```

Population scripts

I recommend you do the population of the database after the triggers have been defined. That way, you will be able to see the audit logs and autofill on the trainer schema.

```
INSERT INTO type VALUES('none', 'none', 'none');
INSERT INTO type VALUES('fire', 'none', 'none');
INSERT INTO type VALUES('water', 'none', 'none');
INSERT INTO type VALUES('grass', 'none', 'none');
INSERT INTO type VALUES('electric', 'none', 'none');
INSERT INTO type VALUES('flying', 'none', 'none');
INSERT INTO type VALUES('poison', 'none', 'none');


UPDATE type
SET weak_against = 'water', strong_against= 'grass'
WHERE type_name='fire';

UPDATE type
SET weak_against = 'fire', strong_against= 'water'
WHERE type_name='grass';

UPDATE type
SET weak_against = 'grass', strong_against= 'fire'
WHERE type_name='water';

UPDATE type
SET weak_against = 'none', strong_against= 'flying'
WHERE type_name='electric';

UPDATE type
SET weak_against = 'electric', strong_against= 'grass'
WHERE type_name='flying';



INSERT INTO pokemon_type(type_1,type_2) VALUES('fire', 'none');
INSERT INTO pokemon_type(type_1,type_2) VALUES('fire', 'flying');
INSERT INTO pokemon_type(type_1,type_2) VALUES('grass', 'none');
INSERT INTO pokemon_type(type_1,type_2) VALUES('grass', 'poison');
INSERT INTO pokemon_type(type_1,type_2) VALUES('water', 'none');
INSERT INTO pokemon_type(type_1,type_2) VALUES('electric', 'none');
INSERT INTO pokemon VALUES(1, 'bulbasaur', 4, 1, 1);
INSERT INTO pokemon VALUES(2, 'ivysaur', 4, 2, 1);
INSERT INTO pokemon VALUES(3, 'venusaur', 4, 3, 1);


INSERT INTO pokemon VALUES(4, 'charmader', 1, 4, 4);
INSERT INTO pokemon VALUES(5, 'charmeleon', 1, 5, 4);
INSERT INTO pokemon VALUES(6, 'charizard', 2, 6, 4);

INSERT INTO pokemon VALUES(7, 'squirtle', 5, 7, 7);
INSERT INTO pokemon VALUES(8, 'wartortle', 5, 8, 7);
INSERT INTO pokemon VALUES(9, 'blastoise', 5, 9, 7);


INSERT INTO pokemon VALUES(172, 'pichu', 6, 172, 172);
INSERT INTO pokemon VALUES(25, 'pikachu', 6, 25, 172);
INSERT INTO pokemon VALUES(26, 'raichu', 6, 26, 172);


INSERT INTO move VALUES('flamethrower', 100 , 100, 'special', 'fire');
INSERT INTO move VALUES('flare blitz', 120 , 100, 'physical', 'fire');
INSERT INTO move VALUES('hydro pump', 120 , 90, 'special', 'water');
INSERT INTO move VALUES('aqua jet', 90 , 100, 'special', 'water');
INSERT INTO move VALUES('razor leaf', 60 , 100, 'special', 'grass');
INSERT INTO move VALUES('vine whip', 50 , 100, 'physical', 'grass');
INSERT INTO move VALUES('thunderbolt', 100 , 100, 'special', 'electric');
INSERT INTO move VALUES('volt tackle', 120 , 100, 'physical', 'electric');
INSERT INTO move VALUES('wind cutter', 100 , 100, 'special', 'flying');
INSERT INTO move VALUES('acrobatics', 100 , 100, 'physcial', 'flying');

INSERT INTO location VALUES('pallet town');
INSERT INTO location VALUES('route 1');
INSERT INTO location VALUES('route 2');
INSERT INTO location VALUES('route 3');
INSERT INTO location VALUES('route 4');
INSERT INTO location VALUES('goldenrod city');
INSERT INTO location VALUES('pokemon league');

INSERT INTO trainer VALUES(2201800462);

INSERT INTO trainer_class VALUES('fire bender','fire');
INSERT INTO trainer_class VALUES('swimmer','water');
INSERT INTO trainer_class VALUES('gardener','grass');
INSERT INTO trainer_class VALUES('bird keeper','flying');
INSERT INTO trainer_class VALUES('electrician','electric');
INSERT INTO can_learn VALUES(1,'vine whip', 20);
INSERT INTO can_learn VALUES(8,'aqua jet', 30);
INSERT INTO can_learn VALUES(9,'aqua jet', 35);
INSERT INTO can_learn VALUES(9,'hydro pump', 49);
INSERT INTO can_learn VALUES(6,'flamethrower', 27);
INSERT INTO can_learn VALUES(6,'flare blitz', 55);
INSERT INTO can_learn VALUES(25,'thunderbolt', 26);
INSERT INTO can_learn VALUES(25,'volt tackle', 66);


INSERT INTO pokemon_is_of_type VALUES(1,'grass');
INSERT INTO pokemon_is_of_type VALUES(2,'grass');
INSERT INTO pokemon_is_of_type VALUES(3,'grass');
INSERT INTO pokemon_is_of_type VALUES(4,'fire');
INSERT INTO pokemon_is_of_type VALUES(5,'fire');
INSERT INTO pokemon_is_of_type VALUES(6,'fire');
INSERT INTO pokemon_is_of_type VALUES(7,'water');
INSERT INTO pokemon_is_of_type VALUES(8,'water');
INSERT INTO pokemon_is_of_type VALUES(9,'water');
INSERT INTO pokemon_is_of_type VALUES(172,'electric');
INSERT INTO pokemon_is_of_type VALUES(25,'electric');
INSERT INTO pokemon_is_of_type VALUES(26,'electric');

INSERT INTO can_be_found_at VALUES(25,'pallet town');
INSERT INTO can_be_found_at VALUES(1,'pallet town');
INSERT INTO can_be_found_at VALUES(4,'pallet town');
INSERT INTO can_be_found_at VALUES(7,'pallet town');
INSERT INTO can_be_found_at VALUES(26,'goldenrod city');
INSERT INTO can_be_found_at VALUES(172,'route 1');
INSERT INTO can_be_found_at VALUES(25,'route 2');
INSERT INTO can_be_found_at VALUES(26,'route 3');
INSERT INTO can_be_found_at VALUES(3,'route 3');
INSERT INTO can_be_found_at VALUES(6,'route 3');
INSERT INTO can_be_found_at VALUES(9,'route 3');

INSERT INTO captured VALUES(25,2201800462,'pallet town',CURDATE());
INSERT INTO captured VALUES(6,2201800462,'route 3',CURDATE());
INSERT INTO captured VALUES(25,2201800462,'route 2',CURDATE());
INSERT INTO captured VALUES(9,2201800462,'route 3',CURDATE());
```

Triggers

Trigger 1:

The trigger given below is used to automatically populate the trainer table using entries in the pokemon table.

```
delimiter $$
CREATE TRIGGER cascade_trainer_lic_no
AFTER INSERT
ON pokemon.pokemon
FOR EACH ROW
BEGIN
   INSERT INTO trainer
      (license_no)
   values(new.trainer_lic_no);
END$$
```

Triggers 2 - 10:

Given below are 9 triggers for creating an audit trail on the pokemon database. There are 3 triggers for each of the 3 major tables. 1 insert, 1 delete and 1 update for tables pokemon, move and type. This way we will be able to know who has updated what table at what time.

```
CREATE TRIGGER pokemon.audit_pokemon_i
AFTER INSERT
ON pokemon.pokemon
FOR EACH ROW
BEGIN
   DECLARE tablenme VARCHAR(255);
   DECLARE operation VARCHAR(255);
   DECLARE username VARCHAR(255);
   DECLARE at_time DATETIME;

   SET username = USER();
   SET at_time = NOW();
   SET tablenme = 'Pokemon';
   SET operation = 'INSERT';

   INSERT INTO audit_logs(table_altered,operation,by_user,at_time)
      VALUES(tablenme, operation,username,at_time);
END$$

CREATE TRIGGER pokemon.audit_pokemon_u
```

```
AFTER INSERT
ON pokemon.pokemon
FOR EACH ROW
BEGIN
   DECLARE tablenme VARCHAR(255);
   DECLARE operation VARCHAR(255);
   DECLARE username VARCHAR(255);
   DECLARE at_time DATETIME;

   SET username = USER();
   SET at_time = NOW();
   SET tablenme = 'Pokemon';
   SET operation = 'UPDATE';

   INSERT INTO audit_logs(table_altered,operation,by_user,at_time)
      VALUES(tablenme, operation,username,at_time);
END$$

CREATE TRIGGER pokemon.audit_pokemon_d
AFTER DELETE
ON pokemon.pokemon
FOR EACH ROW
BEGIN
   DECLARE tablenme VARCHAR(255);
   DECLARE operation VARCHAR(255);
   DECLARE username VARCHAR(255);
   DECLARE at_time DATETIME;

   SET username = USER();
   SET at_time = NOW();
   SET tablenme = 'Pokemon';
   SET operation = 'DELETE';

   INSERT INTO audit_logs(table_altered,operation,by_user,at_time)
      VALUES(tablenme, operation,username,at_time);
END$$



CREATE TRIGGER pokemon.audit_move_i
AFTER INSERT
ON pokemon.move
FOR EACH ROW
BEGIN
   DECLARE tablenme VARCHAR(255);
   DECLARE operation VARCHAR(255);
```

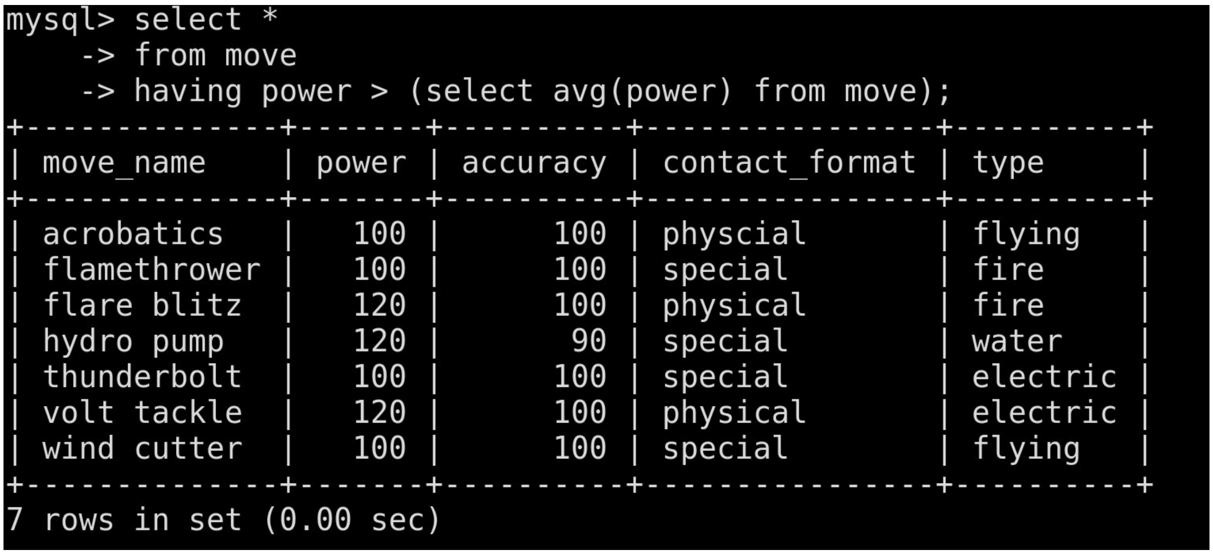
```
    DECLARE username VARCHAR(255);
    DECLARE at_time DATETIME;

    SET username = USER();
    SET at_time = NOW();
    SET tablenme = 'Move';
    SET operation = 'INSERT';

    INSERT INTO audit_logs(table_altered,operation,by_user,at_time)
        VALUES(tablenme, operation,username,at_time);
END$$

CREATE TRIGGER pokemon.audit_move_u
AFTER INSERT
ON pokemon.move
FOR EACH ROW
BEGIN
    DECLARE tablenme VARCHAR(255);
    DECLARE operation VARCHAR(255);
    DECLARE username VARCHAR(255);
    DECLARE at_time DATETIME;

    SET username = USER();
    SET at_time = NOW();
    SET tablenme = 'Move';
    SET operation = 'UPDATE';

    INSERT INTO audit_logs(table_altered,operation,by_user,at_time)
        VALUES(tablenme, operation,username,at_time);
END$$

CREATE TRIGGER pokemon.audit_move_d
AFTER DELETE
ON pokemon.move
FOR EACH ROW
BEGIN
    DECLARE tablenme VARCHAR(255);
    DECLARE operation VARCHAR(255);
    DECLARE username VARCHAR(255);
    DECLARE at_time DATETIME;

    SET username = USER();
    SET at_time = NOW();
    SET tablenme = 'Move';
    SET operation = 'DELETE';
    INSERT INTO audit_logs(table_altered,operation,by_user,at_time)
        VALUES(tablenme, operation,username,at_time);
END$$



CREATE TRIGGER pokemon.audit_type_i
AFTER INSERT
ON pokemon.type
FOR EACH ROW
BEGIN
    DECLARE tablenme VARCHAR(255);
    DECLARE operation VARCHAR(255);
    DECLARE username VARCHAR(255);
    DECLARE at_time DATETIME;

    SET username = USER();
    SET at_time = NOW();
    SET tablenme = 'Type';
    SET operation = 'INSERT';

    INSERT INTO audit_logs(table_altered,operation,by_user,at_time)
        VALUES(tablenme, operation,username,at_time);
END$$

CREATE TRIGGER pokemon.audit_type_u
AFTER INSERT
ON pokemon.type
FOR EACH ROW
BEGIN
    DECLARE tablenme VARCHAR(255);
    DECLARE operation VARCHAR(255);
    DECLARE username VARCHAR(255);
    DECLARE at_time DATETIME;

    SET username = USER();
    SET at_time = NOW();
    SET tablenme = 'type';
    SET operation = 'UPDATE';

    INSERT INTO audit_logs(table_altered,operation,by_user,at_time)
        VALUES(tablenme, operation,username,at_time);
END$$

CREATE TRIGGER pokemon.audit_type_d
AFTER DELETE
```

```
ON pokemon.type
FOR EACH ROW
BEGIN
   DECLARE tablenme VARCHAR(255);
   DECLARE operation VARCHAR(255);
   DECLARE username VARCHAR(255);
   DECLARE at_time DATETIME;

   SET username = USER();
   SET at_time = NOW();
   SET tablenme = 'Type';
   SET operation = 'DELETE';

   INSERT INTO audit_logs(table_altered,operation,by_user,at_time)
      VALUES(tablenme, operation,username,at_time);
END$$
```

SQL Queries

- List all fire type moves with above average power

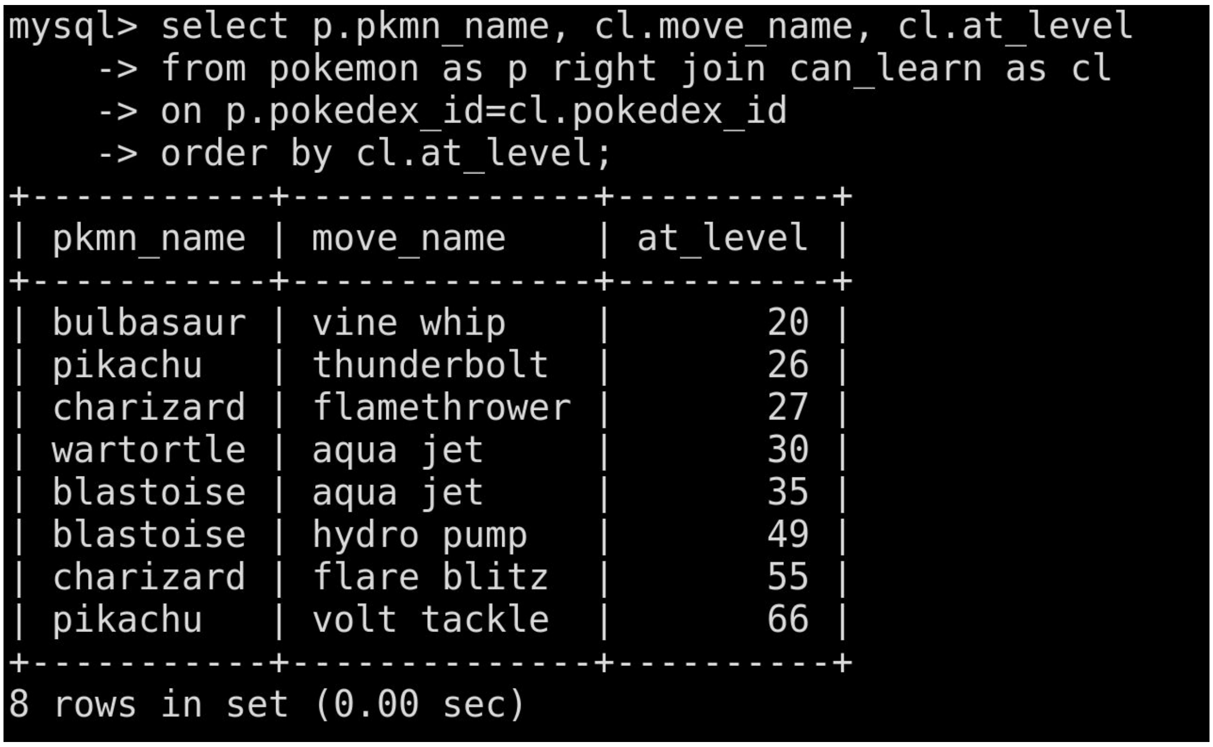
select *
from move
having power > (select avg(power) from move);



```
mysql> select *
    -> from move
    -> having power > (select avg(power) from move);
+-------------+-------+----------+----------------+----------+
| move_name   | power | accuracy | contact_format | type     |
+-------------+-------+----------+----------------+----------+
| acrobatics  |   100 |      100 | physcial       | flying   |
| flamethrower|   100 |      100 | special        | fire     |
| flare blitz |   120 |      100 | physical       | fire     |
| hydro pump  |   120 |       90 | special        | water    |
| thunderbolt |   100 |      100 | special        | electric |
| volt tackle |   120 |      100 | physical       | electric |
| wind cutter |   100 |      100 | special        | flying   |
+-------------+-------+----------+----------------+----------+
7 rows in set (0.00 sec)
```

- List pokemon names and the moves they learn along with the levels in the order of level

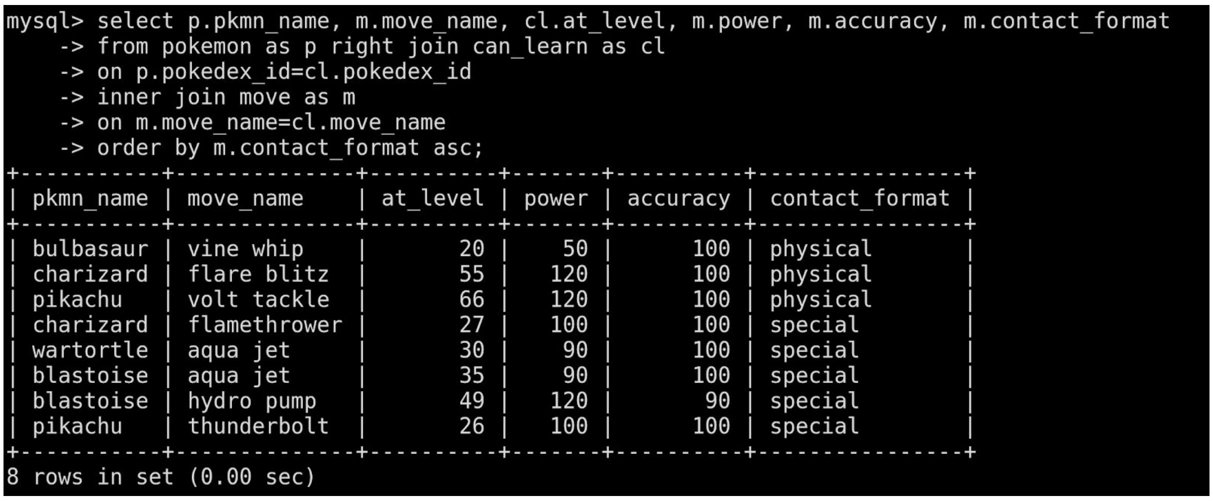
select p.pkmn_name, cl.move_name, cl.at_level
from pokemon as p right join can_learn as cl
on p.pokedex_id=cl.pokedex_id
order by cl.at_level;



```
mysql> select p.pkmn_name, cl.move_name, cl.at_level
    -> from pokemon as p right join can_learn as cl
    -> on p.pokedex_id=cl.pokedex_id
    -> order by cl.at_level;
+-----------+--------------+----------+
| pkmn_name | move_name    | at_level |
+-----------+--------------+----------+
| bulbasaur | vine whip    |       20 |
| pikachu   | thunderbolt  |       26 |
| charizard | flamethrower |       27 |
| wartortle | aqua jet     |       30 |
| blastoise | aqua jet     |       35 |
| blastoise | hydro pump   |       49 |
| charizard | flare blitz  |       55 |
| pikachu   | volt tackle  |       66 |
+-----------+--------------+----------+
8 rows in set (0.00 sec)
```

- List all pokemon and their moves along with move stats (power, accuracy, and contact format)

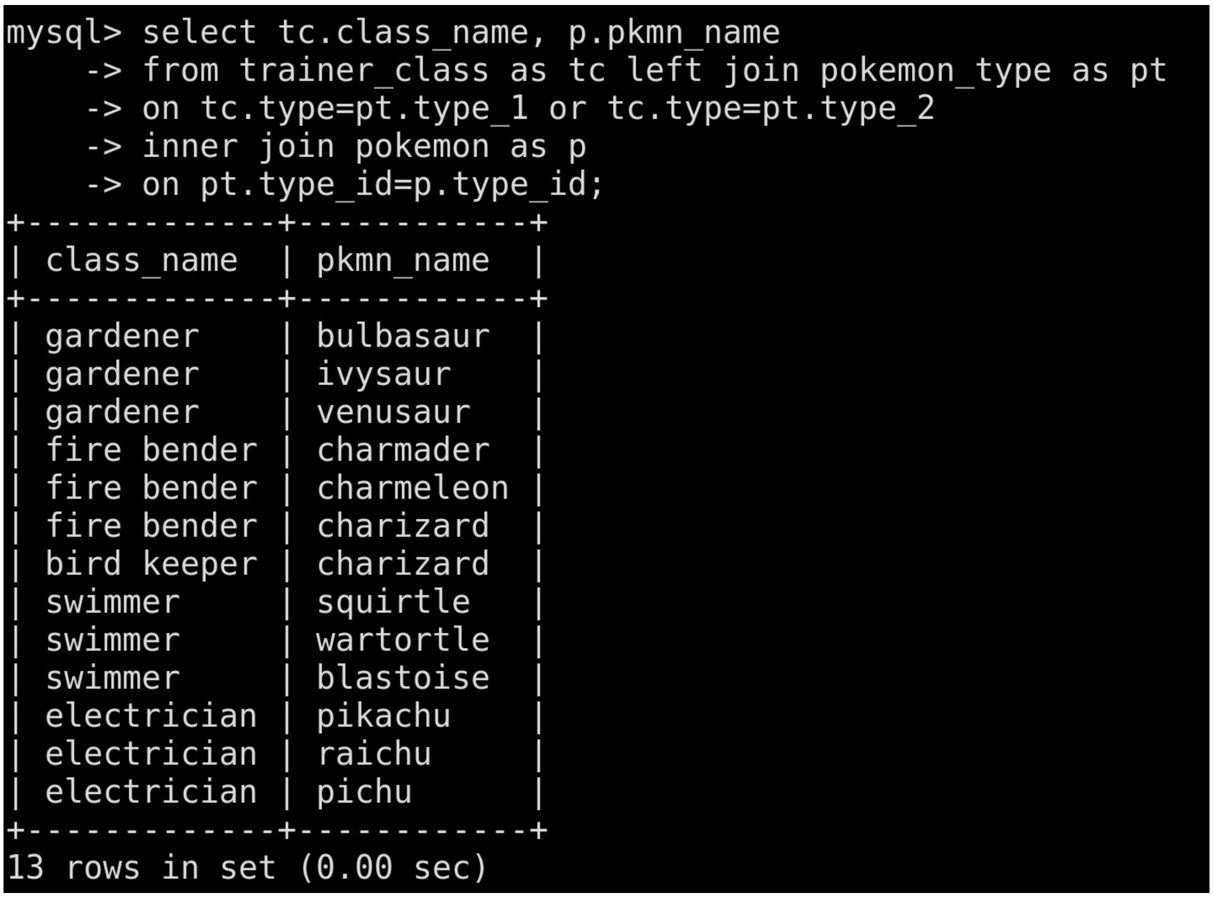
select p.pkmn_name, m.move_name, cl.at_level, m.power, m.accuracy, m.contact_format
from pokemon as p right join can_learn as cl
on p.pokedex_id=cl.pokedex_id
inner join move as m
on m.move_name=cl.move_name
order by m.contact_format asc;



```
mysql> select p.pkmn_name, m.move_name, cl.at_level, m.power, m.accuracy, m.contact_format
    -> from pokemon as p right join can_learn as cl
    -> on p.pokedex_id=cl.pokedex_id
    -> inner join move as m
    -> on m.move_name=cl.move_name
    -> order by m.contact_format asc;
+-----------+--------------+----------+-------+----------+----------------+
| pkmn_name | move_name    | at_level | power | accuracy | contact_format |
+-----------+--------------+----------+-------+----------+----------------+
| bulbasaur | vine whip    |       20 |    50 |      100 | physical       |
| charizard | flare blitz  |       55 |   120 |      100 | physical       |
| pikachu   | volt tackle  |       66 |   120 |      100 | physical       |
| charizard | flamethrower |       27 |   100 |      100 | special        |
| wartortle | aqua jet     |       30 |    90 |      100 | special        |
| blastoise | aqua jet     |       35 |    90 |      100 | special        |
| blastoise | hydro pump   |       49 |   120 |       90 | special        |
| pikachu   | thunderbolt  |       26 |   100 |      100 | special        |
+-----------+--------------+----------+-------+----------+----------------+
8 rows in set (0.00 sec)
```

- List all pokemon that a trainer class can have

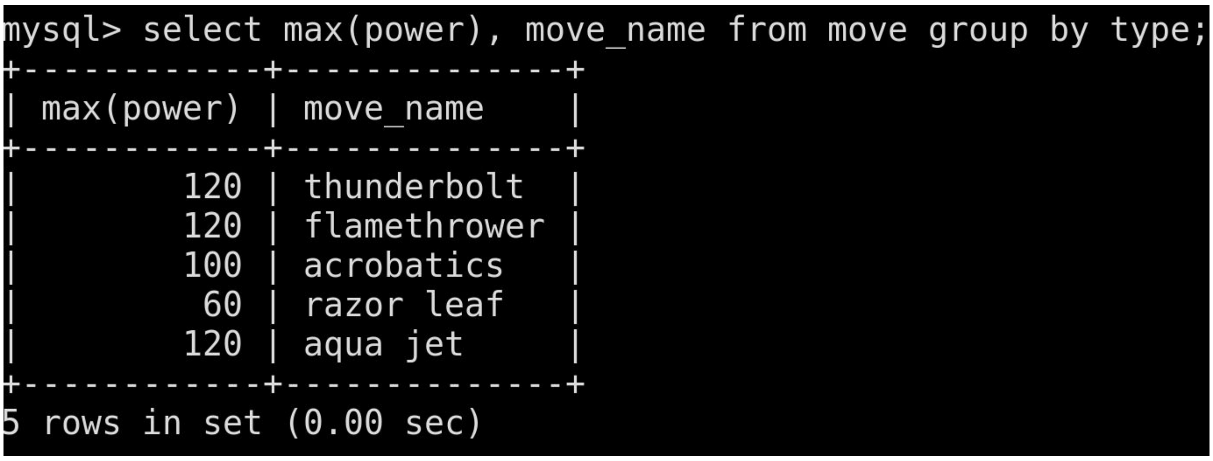
```
select tc.class_name, p.pkmn_name
from trainer_class as tc left join pokemon_type as pt
on tc.type=pt.type_1 or tc.type=pt.type_2
inner join pokemon as p
on pt.type_id=p.type_id;
```



```
mysql> select tc.class_name, p.pkmn_name
    -> from trainer_class as tc left join pokemon_type as pt
    -> on tc.type=pt.type_1 or tc.type=pt.type_2
    -> inner join pokemon as p
    -> on pt.type_id=p.type_id;
+-------------+------------+
| class_name  | pkmn_name  |
+-------------+------------+
| gardener    | bulbasaur  |
| gardener    | ivysaur    |
| gardener    | venusaur   |
| fire bender | charmader  |
| fire bender | charmeleon |
| fire bender | charizard  |
| bird keeper | charizard  |
| swimmer     | squirtle   |
| swimmer     | wartortle  |
| swimmer     | blastoise  |
| electrician | pikachu    |
| electrician | raichu     |
| electrician | pichu      |
+-------------+------------+
13 rows in set (0.00 sec)
```

- List the strongest move wrt power for every type for which a move exists

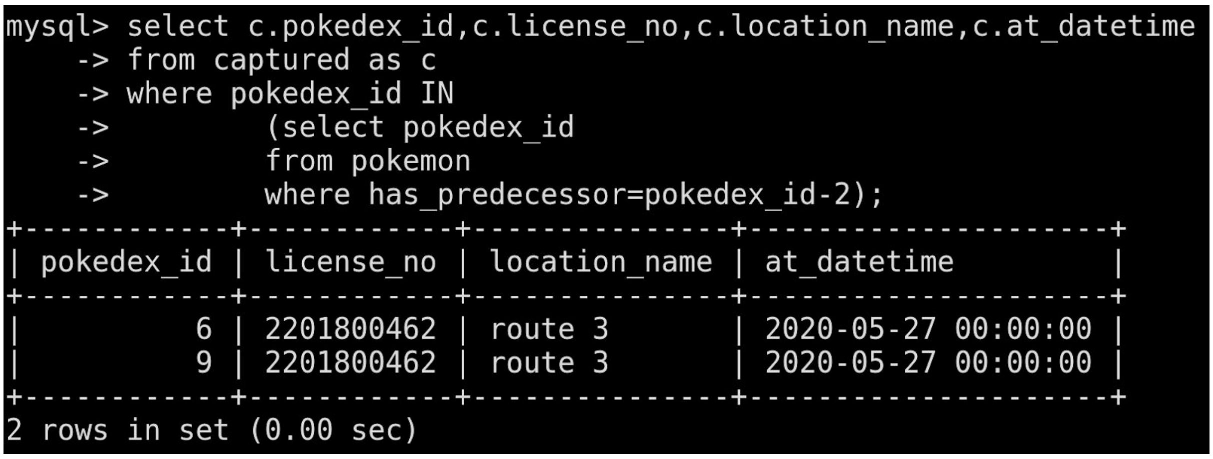
```
select max(power), move_name from move group by type;
```



```
mysql> select max(power), move_name from move group by type;
+------------+--------------+
| max(power) | move_name    |
+------------+--------------+
|        120 | thunderbolt  |
|        120 | flamethrower |
|        100 | acrobatics   |
|         60 | razor leaf   |
|        120 | aqua jet     |
+------------+--------------+
5 rows in set (0.00 sec)
```

- List the second stage pokemon captured and their capture details

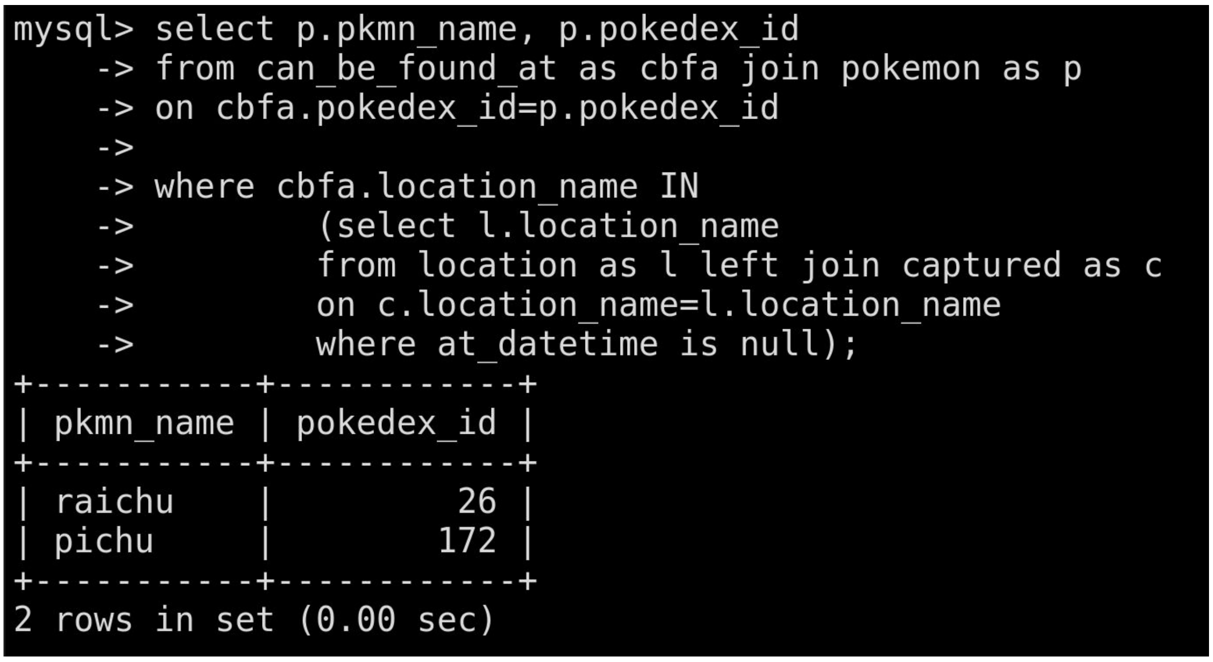
```
select c.pokedex_id,c.license_no,c.location_name,c.at_datetime
from captured as c
where pokedex_id IN
        (select pokedex_id
        from pokemon
        where has_predecessor=pokedex_id-2);
```



```
mysql> select c.pokedex_id,c.license_no,c.location_name,c.at_datetime
    -> from captured as c
    -> where pokedex_id IN
    ->         (select pokedex_id
    ->         from pokemon
    ->         where has_predecessor=pokedex_id-2);
+------------+------------+---------------+---------------------+
| pokedex_id | license_no | location_name | at_datetime         |
+------------+------------+---------------+---------------------+
|          6 | 2201800462 | route 3       | 2020-05-27 00:00:00 |
|          9 | 2201800462 | route 3       | 2020-05-27 00:00:00 |
+------------+------------+---------------+---------------------+
2 rows in set (0.00 sec)
```

- List pokemon that appear in locations that a capture has not occurred yet in

```
select p.pkmn_name, p.pokedex_id
from can_be_found_at as cbfa join pokemon as p
on cbfa.pokedex_id=p.pokedex_id

where cbfa.location_name IN
        (select l.location_name
        from location as l left join captured as c
        on c.location_name=l.location_name
        where at_datetime is null);
```


```
mysql> select p.pkmn_name, p.pokedex_id
    -> from can_be_found_at as cbfa join pokemon as p
    -> on cbfa.pokedex_id=p.pokedex_id
    ->
    -> where cbfa.location_name IN
    ->         (select l.location_name
    ->         from location as l left join captured as c
    ->         on c.location_name=l.location_name
    ->         where at_datetime is null);
+-----------+------------+
| pkmn_name | pokedex_id |
+-----------+------------+
| raichu    |         26 |
| pichu     |        172 |
+-----------+------------+
2 rows in set (0.00 sec)
```

Conclusion

The pokemon DB is a system capable of generating virtually any information that a casual or competitive pokemon player could possibly want. It is easy to maintain, understand and completely airtight against dependency errors occurring.

A few of the limitations are as follows:

- It is difficult to begin data entry as the type table is the root of the DML phase and has 2 FKs pointing to itself. This requires us to input the same value and then alter it later.
- Pokemon_type and pokemon_is_of_type tables are redundant. I didn't catch this glaring mistake until it was too late, but this can definitely be fixed.
- 3 relations have single attributes because including other attributes would have unnecessarily complicated things and taken this project out of scope. In a larger version , I would definitely encourage the people building the DB to include more information such as egg groups, EVs, IVs, Generational classification and so on.

The enhancements are just to make the database larger. As mentioned above, to remove remaining redundancy, possibly normalize more and include more relations to make the database more robust.

A script for ddl, population, triggers and queries combined can be found at
https://github.com/Venkatavaradan-R/PokemonDB