

# **PokéBase**

## **A Centralized Pokédex**



**Designed by  
Michael Read**

## **Table of Contents**

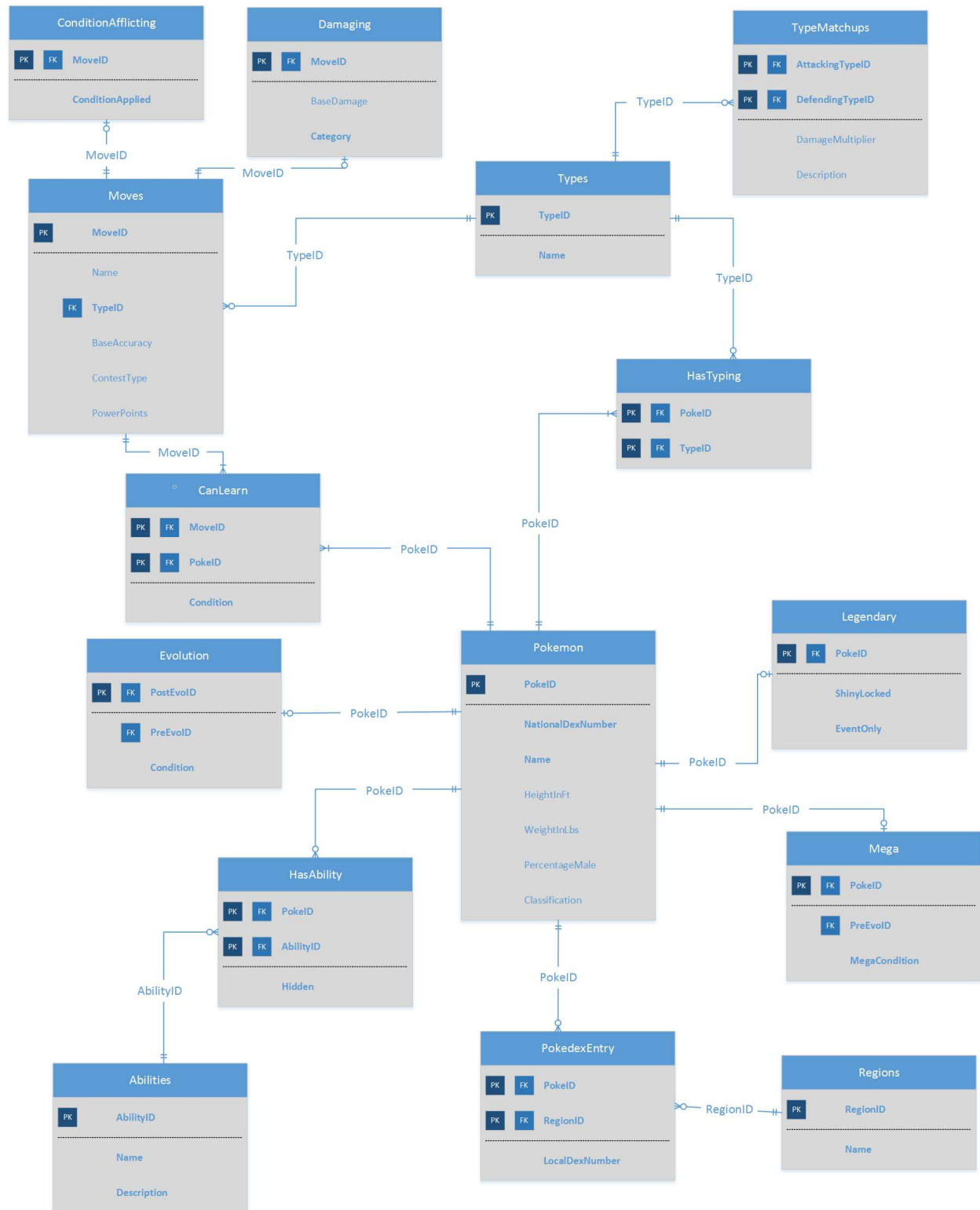
Executive Summary.....	3
E-R Diagram.....	4
Tables.....	5
Views.....	14
Reports and Interesting Queries.....	15
Stored Procedures.....	17
Triggers.....	19
Security.....	21
Implementation Notes.....	22
Known Problems and Future Enhancements.....	23

## **Executive Summary**

The vast world of Pokémon is inhabited by an enormous number of Pokémon, each one different and unique, found in all corners of the globe. From the grassy plains to the cold arctic, from the vast oceans to the skies above, Pokémon can be found almost everywhere. There are even reports of Pokémon not of this world. Due to the incredible numbers of Pokémon that have been discovered, there comes a need for a centralized database of all of this constantly updating information. There are a few around the world who have attempted to fill up their own “Pokédex,” however it would be much more beneficial to everyone around the world to have a single place to find information about any Pokémon, instead of having hundreds of people all attempting to fill up their own version of the same list of Pokémon.

This database would be designed to be used mostly by the trainers around the world discovering all of these new species of Pokémon, as well as the professors in each region, who will be in charge of managing, updating, and doing the important research on all of these new species. The trainers would be in charge of creating the records of all the new Pokémon they find, with only the barest of information about them. It would be the job of the professors to find out all of the details about these new species. The goal of the database will be to help people all around the world to develop their skills as trainers, and allow them to more easily track down the Pokémon which they find interesting, or which they want to add to their teams.

# E-R Diagram



## Tables

**Pokemon** – The Pokemon table holds all of the pokemon, and all the basic information about them.

```
create table Pokemon(
    PokeID          serial not null,
    NationalDexNumber int    not null,
    name            text     not null,
    HeightInFt      decimal,
    WeightInLbs     decimal,
    PercentageMale   decimal,
    Classification   text,
    primary key (PokeID)
);
```

pokeid integer	nationaldexnumber integer	name text	heightinf numeric	weightinlbs numeric	percentagemale numeric	classification text
1	1	Bulbasaur	2.33	15.2	0.875	The Seed Pokemon
2	2	Ivysaur	3.25	28.7	0.875	The Seed Pokemon
3	3	Venusaur	6.58	220.5	0.875	The Seed Pokemon
4	4	Charmander	2.00	18.7	0.875	The Lizard Pokemon
5	5	Charmeleon	3.58	41.9	0.875	The Flame Pokemon
6	6	Charizard	5.58	199.5	0.875	The Flame Pokemon
7	7	Squirtle	1.66	19.8	0.875	The Tiny Turtle Pokemon
8	8	Wartortle	3.25	49.6	0.875	The Turtle Pokemon
9	9	Blastoise	5.25	188.5	0.875	The Shellfish Pokemon
10	3	Mega Venusaur	7.83	342.8	0.875	The Seed Pokemon
11	6	Mega Charizard X	5.58	243.6	0.875	The Flame Pokemon
12	6	Mega Charizard Y	5.58	221.6	0.875	The Flame Pokemon
13	9	Mega Blastoise	5.25	222.9	0.875	The Shellfish Pokemon
14	648	Meloetta	2.00	14.3	<NULL>	The Melody Pokemon
15	251	Celebi	2.00	11.0	<NULL>	The Time Travel Pokemon
16	801	Magearna	3.25	177.5	<NULL>	The Artificial Pokemon
17	374	Beldum	2.00	209.9	<NULL>	The Iron Ball Pokemon
18	375	Metang	3.92	446.4	<NULL>	The Iron Claw Pokemon
19	376	Metagross	5.25	1212.5	<NULL>	The Iron Leg Pokemon
20	376	Mega Metagross	8.17	2078.7	<NULL>	The Iron Leg Pokemon
21	255	Torchic	1.33	5.5	0.875	The Chick Pokemon
22	256	Combusken	2.92	43.0	0.875	The Young Fowl Pokemon
23	257	Blaziken	6.25	114.6	0.875	The Blaze Pokemon
24	257	Mega Blaziken	6.25	114.6	0.875	The Blaze Pokemon

## Functional Dependencies –

PokeID → NationalDexNumber, Name, HeightInFt,  
WeightInLbs, PercentageMale, Classification

**Legendary** – Subtype of Pokemon. Holds basic information pertaining to legendary pokémon, of which only a few of each species exist

```
create table Legendary (
    PokeID      int      not null references Pokemon(PokeID),
    ShinyLocked boolean not null,
    EventOnly   boolean not null,
    Primary key (PokeID)
);
```

pokeid integer	shinylocked boolean	eventonly boolean
14	t	t
15	t	t
16	t	t

### Functional Dependencies –

PokeID → ShinyLocked, EventOnly

**Mega** – Subtype of Pokemon. Holds basic information pertaining to Mega pokémon

```
create table Mega (
    PokeID      int  not null references Pokemon(PokeID),
    PreEvoID    int  not null references Pokemon(PokeID),
    MegaCondition text not null,
    primary key (PokeID)
);
```

pokeid integer	preevoid integer	megacondition text
10	3	Holding a Venusaurite
11	6	Holding a Charizardite X
12	6	Holding a Charizardite Y
13	9	Holding a Blastoisinite
20	19	Holding a Metagrossite
24	23	Holding a Blazikenite

### Functional Dependencies –

PokeID → PreEvoID,  
MegaCondition

**Evolution** – Subtype of Pokemon. Holds basic information pertaining to an evolved pokémon, and describes what must be done before it can evolve.

```
create table Evolution (
    PokeID      int not null references Pokemon(PokeID),
    PreEvoID    int not null references Pokemon(PokeID),
    Condition   text not null,
    primary key (PokeID)
);
```

pokeid integer	preevoid integer	condition text
2	1	Level 16
3	2	Level 32
5	4	Level 16
6	5	Level 36
8	7	Level 16
9	8	Level 36
18	17	Level 20
19	18	Level 45
22	21	Level 16
23	22	Level 36

## Functional Dependencies –

PokeID → PreEvoID, Condition

**Regions** – Table containing the basic information about the different regions in the world.

```
create table Regions (
    RegionID    serial not null,
    Name        text not null,
    primary key (RegionID)
);
```

regionid integer	name text
1	Kanto
2	Johto
3	Hoenn
4	Alola

## Functional Dependencies –

RegionID → Name

## PokedexEntry – Used to link Pokémon to the different regions.

```
create table PokedexEntry (
    PokeID    int not null references Pokemon(PokeID),
    RegionID  int not null references Regions(RegionID),
    LocalDexNumber int not null,
    primary key (PokeID, RegionID)
);
```

pokeid integer	regionid integer	localdexnumber integer
17	4	214
18	4	215
19	4	216
20	4	216
17	3	190
18	3	191
19	3	192
20	3	192
21	3	4
22	3	5
23	3	6
24	3	6
1	1	1
2	1	2
3	1	3
4	1	4
5	1	5
6	1	6
7	1	7
8	1	8
9	1	9
10	1	3
11	1	6
12	1	6
13	1	9
15	2	251
15	3	386
16	4	301

## Functional Dependencies –

PokeID, RegionID → LocalDexNumber

## Abilities – Table containing the basic information pertaining to the different abilities pokémon can have.

```
create table Abilities (
    AbilityID    serial not null,
    Name          text    not null,
    Description text    not null,
    primary key (AbilityID)
);
```

abilityid integer	name text	description text
1	Blaze	When HP is below 1/3rd its maximum, power of Fire-type moves is increased by 50%
2	Torrent	When HP is below 1/3rd its maximum, power of Water-type moves is increased by 50%
3	Overgrow	When HP is below 1/3rd its maximum, power of Grass-type moves is increased by 50%
4	Chlorophyll	When sunny, the Pokémon's Speed doubles. However, Speed will not double on the turn weather becomes Strong Sunlight
5	Clear Body	Opponents moves which lower this Pokémon's stats have no effect. However this Pokémon may lower its own stats with its own moves
6	Light Metal	Halves the Pokémon's weight
7	Speed Boost	Speed increases by one stage at the end of each turn.
8	Tough Claws	Increases the power of moves that make physical contact by 33%

## Functional Dependencies –

AbilityID → Description



**HasAbility** – Used to connect the different pokémon to the abilities they can have, and whether that ability is a hidden, or extremely rare, ability for the pokémon.

```
create table HasAbility (
    PokeID    int not null references Pokemon(PokeID),
    AbilityID int not null references Abilities(AbilityID),
    Hidden    boolean not null,
    primary key (PokeID, AbilityID)
);
```

### Functional Dependencies –

PokeID, AbilityID → Hidden

**Types** – Table that has the basic information about the different pokémon types.

```
create table Types (
    TypeID serial not null,
    Name    text not null,
    primary key (TypeID)
);
```

### Functional Dependencies –

TypeID → Name

pokeid integer	abilityid integer	hidden boolean
1	3	f
2	3	f
3	3	f
4	1	f
5	1	f
6	1	f
7	2	f
8	2	f
9	2	f
11	8	f
1	4	t
2	4	t
3	4	t
17	5	f
17	6	t
18	5	f
18	6	t
19	5	f
19	6	t
20	8	f
21	1	f
21	7	t
22	1	f
22	7	t
23	1	f
23	7	t
24	7	f

typeid integer	name text
1	Psychic
2	Steel
3	Fire
4	Fighting
5	Grass
6	Poison
7	Flying
8	Water
9	Fairy
10	Dragon
11	Normal

**HasTyping** – Table that links the different pokémon to their typings.

```
create table HasTyping (
    PokeID int not null references Pokemon(PokeID),
    TypeID int not null references Types(TypeID),
    primary key (PokeID, TypeID)
);
```

**Functional Dependencies –**

PokeID, TypeID →

pokeid integer	typeid integer
1	5
1	6
2	5
2	6
3	5
3	6
4	3
5	3
6	3
6	7
7	8
8	8
9	8
10	5
10	6
11	3
11	10
12	3
12	7
13	8
14	1
14	11
21	3
22	3
22	4
23	3
23	4
24	3
24	4

**TypeMatchup** – Table that describes how different typings fare against each other, whether they are effective to use against certain other typings or not.

```
create table TypeMatchup (
    AttackingTypeID int not null references Types(TypeID),
    DefendingTypeID int not null references Types(TypeID),
    DamageMultiplier decimal,
    Description text check (Description in ('Its super effective!',
    'Its normally effective', 'Its not very effective', 'No effect')),
    primary key (AttackingTypeID, DefendingTypeID)
);
```

## Functional Dependencies –

AttackingTypeID, DefendingTypeID → DamageMultiplier,  
Description

attackingtypeid integer	defendingtypeid integer	damagemultiplier numeric	description text
1	1	0.5	Its not very effective
1	4	2	Its super effective!
3	5	2	Its super effective!
5	8	2	Its super effective!
8	3	2	Its super effective!
3	8	0.5	Its not very effective
8	5	0.5	Its not very effective
5	3	0.5	Its not very effective
10	10	2	Its super effective!
9	10	2	Its super effective!
10	9	0	No effect
7	5	2	Its super effective!
1	3	1	Its normally effective
4	3	1	Its normally effective
1	2	0.5	Its not very effective
2	1	1	Its normally effective

**Moves** – Table that holds all of the moves that we have found out pokémon can use.

```
create table Moves (
    MoveID serial not null,
    Name text not null,
    TypeID int not null references Types(TypeID),
    BaseAccuracy decimal,
    ContestType text check (ContestType in ('Beauty', 'Cool', 'Cute',
'Smart', 'Tough')),
    PowerPoints int,
    primary key (MoveID)
);
```

moveid integer	name text	typeid integer	baseaccuracy numeric	contesttype text	powerpoints integer
1	Tackle	11	1	Tough	35
2	Water Gun	8	1	Cute	25
3	Scratch	11	1	Tough	35
4	Growl	11	1	Cute	40
5	Ember	3	1	Cute	25
6	Take Down	11	0.85	Tough	20
7	Poison Powder	6	0.75	Clever	35
8	Vine Whip	5	1	Cool	25

## Functional Dependencies –

MoveID → TypeID, BaseAccuracy, ContestType, PowerPoints

**Damaging** – Subtype of Moves. Holds the information regarding moves that deal damage to the other pokémon.

```
create table Damaging (
    MoveID int not null references Moves(MoveID),
    BaseDamage int,
    Category text not null check (Category in ('Physical', 'Special')),
    primary key (MoveID)
);
```

## Functional Dependencies –

MoveID → BaseDamage, Category

moveid integer	basedamage integer	category text
1	50	Physical
2	40	Special
3	40	Physical
5	40	Special
6	90	Physical
8	45	Physical

**ConditionAfflicting** – Subtype of Moves. Holds the information regarding moves that afflict a status or condition either pokémon involved in the battle.

```
create table ConditionAfflicting (
    MoveID int not null references Moves(MoveID),
    ConditionApplied text not null,
    primary key (MoveID)
);
```

moveid integer	conditionapplied text
4	Lowers opponents attack by one stage
5	May induce burn
6	User takes recoil damage equal to 25% of the damage inflicted
7	Induces poison

## Functional Dependencies –

MoveID → ConditionApplied

**CanLearn** – Connects pokémon to the moves that they can learn, and describes what must be achieved for the pokémon to learn the move.

```
create table CanLearn (
    PokeID int not null references Pokemon(PokeID),
    MoveID int not null references Moves(MoveID),
    Condition text not null,
    primary key (PokeID, MoveID)
);
```

## Functional Dependencies –

PokeID, MoveID → Condition

pokeid integer	moveid integer	condition text
1	1	Level 1
2	1	Level 1
3	1	Level 1
4	3	Level 1
5	3	Level 1
6	3	Level 1
7	1	Level 1
8	1	Level 1
9	1	Level 1
7	2	Level 7
8	2	Level 7
9	2	Level 7
21	3	Level 1
21	4	Level 1
22	3	Level 1
22	4	Level 1
23	3	Level 1
23	4	Level 1
1	7	Level 13
2	7	Level 13
3	7	Level 13
17	6	Level 1
18	6	Level 1
19	6	Level 1

# Views

## EvolutionPokemon – Shows all of the pokémon that are involved in evolution

```
create view EvolutionPokemon as
```

```
select p.PokeID, p.NationalDexNumber, p.Name, p.HeightInFt,
p.WeightInLbs, p.PercentageMale, p.Classification, e.Condition as
EvolvedCondition, o.name as EvolvedFrom, v.Condition EvolvesCondition,
k.name as EvolvesInto
```

```
from Pokemon p full outer join Evolution e on p.PokeID = e.PokeID
```

```
left outer join Pokemon o on e.preEvoID = o.PokeID
```

```
full outer join Evolution v on p.PokeID = v.PreEvoID
```

```
left outer join Pokemon k on v.PokeID = k.PokeID
```

```
where (o.name is not null or k.name is not null)
```

```
order by NationalDexNumber asc;
```

pokeid integer	nationaldexnumber integer	name text	heightinft numeric	weightinlbs numeric	percentagemale numeric	classification text	evolvedcondition text	evolvedfrom text	evolvescondition text	evolvesinto text
1	1	Bulbasaur	2.33	15.2	0.875	The Seed Pokemon	<NULL>	<NULL>	Level 16	Ivysaur
2	2	Ivysaur	3.25	28.7	0.875	The Seed Pokemon	Level 16	Bulbasaur	Level 32	Venusaur
3	3	Venusaur	6.58	220.5	0.875	The Seed Pokemon	Level 32	Ivysaur	<NULL>	<NULL>
4	4	Charmander	2.00	18.7	0.875	The Lizard Pokemon	<NULL>	<NULL>	Level 16	Charmeleon
5	5	Charmeleon	3.58	41.9	0.875	The Flame Pokemon	Level 16	Charmander	Level 36	Charizard
6	6	Charizard	5.58	199.5	0.875	The Flame Pokemon	Level 36	Charmeleon	<NULL>	<NULL>
7	7	Squirtle	1.66	19.8	0.875	The Tiny Turtle Pokemon	<NULL>	<NULL>	Level 16	Wartortle
8	8	Wartortle	3.25	49.6	0.875	The Turtle Pokemon	Level 16	Squirtle	Level 36	Blastoise
9	9	Blastoise	5.25	188.5	0.875	The Shellfish Pokemon	Level 36	Wartortle	<NULL>	<NULL>
21	255	Torchic	1.33	5.5	0.875	The Chick Pokemon	<NULL>	<NULL>	Level 16	Combusken
22	256	Combusken	2.92	43.0	0.875	The Young Fowl Pokemon	Level 16	Torchic	Level 36	Blaziken
23	257	Blaziken	6.25	114.6	0.875	The Blaze Pokemon	Level 36	Combusken	<NULL>	<NULL>
17	374	Beldum	2.00	209.9	<NULL>	The Iron Ball Pokemon	<NULL>	<NULL>	Level 20	Metang
18	375	Metang	3.92	446.4	<NULL>	The Iron Claw Pokemon	Level 20	Beldum	Level 45	Metagross
19	376	Metagross	5.25	1212.5	<NULL>	The Iron Leg Pokemon	Level 45	Metang	<NULL>	<NULL>

## Reports and Interesting Queries

Query to return the number of different pokemon that can be found in each of the different regions

```
select r.name, count(r.name)
from Pokemon p inner join PokedexEntry pe on p.PokeID = pe.PokeID
            inner join regions r          on r.RegionID = pe.RegionID
group by (r.name);
```

name text	count bigint
Johto	1
Hoenn	9
Kanto	13
Alola	5

Query to return all the pokémon that are dual-type

```
select name
from Pokemon p inner join hasTyping ht on p.PokeID = ht.PokeID
group by (name)
having count(name) = 2;
```

name text
Mega Blaziken
Charizard
Ivysaur
Mega Venusaur
Combusken
Meloetta
Venusaur
Mega Charizard X
Blaziken
Bulbasaur
Mega Charizard Y

Query to return the typing of which the most pokémon have

```
select name
from types t inner join hasTyping ht on t.TypeID = ht.TypeID
group by (name)
order by count(name) desc
limit 1;
```

name
text
Fire



## **Stored Procedures**

**check\_typings** – Returns a trigger which stops a user from giving a pokémon more than two typings (See Triggers, 19)

```
create or replace function check_typings() returns trigger
as
$$
declare
    rc int;
begin
    select count(*) into rc
        from hasTyping
        where PokeID = NEW.PokeID;
    if (rc < 2)
        then
            /*All good. Do nothing. Move along. These are not the droids
            you're looking for*/
        else
            raise exception 'A pokemon cannot have more than two types';
        end if;
    return new;
end;
$$
language PLPGSQL;
```

**create\_type\_matchups** – Returns a trigger which automatically adds in all of the possible new type matchups when someone creates a new typing, with the new type as the attacking type (See Triggers, 20)

```
create or replace function create_type_matchups() returns trigger
as
$$
declare
    defendingType record;
begin
    FOR defendingType IN select TypeID from Types LOOP

        insert into TypeMatchup (AttackingTypeID, DefendingTypeID)
            values (new.TypeID, defendingType.TypeID);

    end loop;
    return null;
end;
$$
language PLPGSQL;
```

## Triggers

**manage\_pokemon typings** – Before making an insert or update to the hasTyping table, this executes the check\_typings procedure (See Stored Procedures, 17)

```
create trigger manage_pokemon_typings
  before insert or update
  on hasTyping
  for each row
  execute procedure check_typings();
```

```
ERROR:  A pokemon cannot have more than two types
***** Error *****
```

```
ERROR: A pokemon cannot have more than two types
SQL state: P0001
```

**manage\_type\_matchups** – When a new entry is added to the Types table, this calls the create\_type\_matchups procedure (See Stored Procedures, 18)

```
create trigger manage_type_matchups
after insert
on Types
for each row
execute procedure create_type_matchups();
```

**Before:**

attackingtypeid integer	defendingtypeid integer	damagemultiplier numeric	description text
1	1	0.5	Its not very effective
1	4	2	Its super effective!
3	5	2	Its super effective!
5	8	2	Its super effective!
8	3	2	Its super effective!
3	8	0.5	Its not very effective
8	5	0.5	Its not very effective
5	3	0.5	Its not very effective
10	10	2	Its super effective!
9	10	2	Its super effective!
10	9	0	No effect
7	5	2	Its super effective!
1	3	1	Its normally effective
4	3	1	Its normally effective
1	2	0.5	Its not very effective
2	1	1	Its normally effective

**After:**

attackingtypeid integer	defendingtypeid integer	damagemultiplier numeric	description text
1	1	0.5	Its not very effective
1	4	2	Its super effective!
3	5	2	Its super effective!
5	8	2	Its super effective!
8	3	2	Its super effective!
3	8	0.5	Its not very effective
8	5	0.5	Its not very effective
5	3	0.5	Its not very effective
10	10	2	Its super effective!
9	10	2	Its super effective!
10	9	0	No effect
7	5	2	Its super effective!
1	3	1	Its normally effective
4	3	1	Its normally effective
1	2	0.5	Its not very effective
2	1	1	Its normally effective
13	1	<NULL>	<NULL>
13	2	<NULL>	<NULL>
13	3	<NULL>	<NULL>
13	4	<NULL>	<NULL>
13	5	<NULL>	<NULL>
13	6	<NULL>	<NULL>
13	7	<NULL>	<NULL>
13	8	<NULL>	<NULL>
13	9	<NULL>	<NULL>
13	10	<NULL>	<NULL>
13	11	<NULL>	<NULL>
13	13	<NULL>	<NULL>

## **Security**

**Professor Role** – The professors, who have full access to everything, as they are the people who are doing the research.

```
create role professor;  
grant all on all tables in schema public to professor;
```

**Trainer Role** – The trainers, who have the ability to create new pokémon, types, and moves that they find, and look up anything.

```
create role trainer;  
revoke all on all tables in schema public from trainers;  
grant select on all tables in schema public to trainers;  
grant insert on Pokemon, Types, Moves to trainers;
```

**Youngster Role** – The youngsters, regular people who are not out finding new pokemon, types, and moves. They have the ability to see all the data, but cannot change anything.

```
create role youngster;  
revoke all on all tables in schema public from youngster;  
grant select on all tables in schema public to youngster;
```

## **Implementation Notes**

- Since there are pokémon that exist which do not have a gender, when that occurs the PercentageMale field is left null
- Since a pokémon can evolve into multiple different pokémon depending on what conditions are met first, but a pokémon can only have one specific pre-evolution, the ID of the pokémon evolving is not used as a primary key in the Evolution table
- Since Mega-evolution is different to regular evolution in many ways, and since it is important for certain items to know whether a pokémon can evolve further, not including mega-evolution, the Mega and Evolution tables are two separate tables
- When the type matchups are created from the manage\_type\_matchups trigger, it will be necessary to fill in the information for the effectiveness of those matchups, whether they are super effective, normally effective, not very effective, or no effect

## **Known Problems and Future Enhancements**

- There is an instance where a pokémon can evolve into multiple pokémon at the same time, which is currently not supported by this system
- There is an instance where a move has multiple typings used to determine the damage dealt, which is currently not supported by this system
- In the future, there are more tables which can be added, for example an items table which describes all of the different items and what they do and how much they cost/sell for. With that new table added, another linking table could be added to show what pokémon can be found with certain items in their possession
- I would also like to add the stats of the pokémon in the future, as well as tables which describe the different tiers of competitive play, and what pokémon are allowed in each tier. This would allow not just pokémon enthusiasts, but competitive trainers as well to utilize the database and better design their competitive teams