Databases project

# *PokéSallianWorld 2022-2023 – Stage 1*


**List of members:**


| | |
|---|---|
| Valeria Ezquerra Rodriguez | valeria.ezquerra@students.salle.url.edu |
| Eugènia Pacheco i Ferrando | eugenia.pacheco@students.salle.url.edu |
| Sebastián Felix Gorga | sebastianfelix.gorga@students.salle.url.edu |
| Marc Soto José | m.soto@students.salle.url.edu |


**Date of finalization:** 19/05/2023

# Index

# 1 Introduction

This project focuses on the design of the database for the “PokéSallianWorld”, a game based on Pokémon which includes the first four generations with some specific changes. The game is quite complex, including the idea of interaction between trainers and Pokémon, the battles between Pokémon teams and the exploration of the whole Pokémon world, among other features. This monumental game would only be able to be completed if there is a strong foundation in the form of its database.

In this initial phase of the project, our team has been tasked with the design of both the conceptual and relational models for the database of the game, which were splitted in four different blocks which will have their own section in this report.

The team consists of four members, Valèria Ezquerra, Eugènia Pacheco, Marc Soto, and Sebastián Gorga. The tasks were not divided between the members initially, since we wanted each of us to analyze the functionalities and requirements of the game so that everyone was able to comprehend the software in its entirety. Therefore, each member was tasked with creating a draft of the conceptual model (including the four modules) so that we could share and compare between each other the solution implementations that each of us thought about. Through sharing and debating, we were able to complete a final version of the conceptual model and everyone had a strong understanding of the project. Once this was completed, Valèria and Eugènia were tasked with the translation of the conceptual mode into the relational model while Marc and Sebastián worked on the documentation of the project in the form of the report.

# 2 Entity-relationship model

It can be safely said that this was not a simple task since there are plenty of features to consider. This resulted in the creation of a total of thirty-four entities each with their own interconnected relationships.

The model has 3 core entities in its center, the ***Pokémon Species***, the ***Pokémon Creatures*** and the ***Trainers*** and basically everything else stems from these three entities and go on to create the conceptual model that we have designed.

Even though the functionalities and requirements that we were provided with were divided into 4 modules, all of these features are deeply intertwined and in some cases an entity central to one of the models can be related to another one far away in the logic of another module. That is why the four modules are not easily distinguishable in the model itself since everything is related in some form or another with many other features.

We have made use of every tool that has been taught to us including binary, ternary and reflexive relationships, all the different cardinalities and the generalization/specialization of entities. Many attributes have been assigned to the entities while others were included in the relationships themselves when the possibility allowed for it in N to M relationships. With this, we have been able to provide a comprehensive solution to the task at hand.

## 2.1 Diagram

In the following page you will find a screenshot of our conceptual model diagram. If you have any troubles reading it, please access the following link:

https://drive.google.com/file/d/1Ufi5FzJ8OX_BfFM85-Hh3bs2VEH0V_GA/view?usp=sharing

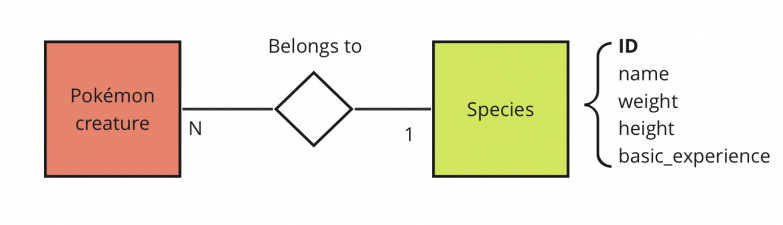
COLOR CODE

GREEN - 1st block
RED - 2nd block
YELLOW - 3rd block
BLUE - 4th block

BLACK - relationships between entities of different blocks

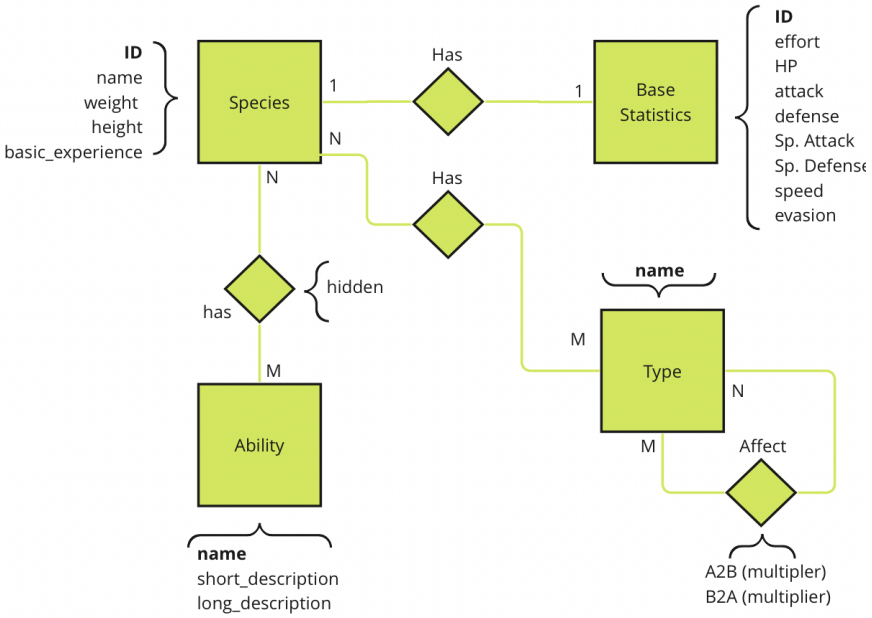
Take place
gift
Subarea
ID
name
Has
Happen
Gym
medal
Leads
Gym leader
Shop
num_floors
sell
stock
discount
Flavour
name
strength
Has
Berries
growing_time
maximum_amount
size
softness
firmness
Stat Enhancer
stat_to_double
time_period
Healing Item
HP_recovered
status_heal
revive?
HP_percentage
Loot
collector_price
Pokéball
max_catch_rate
min_catch_rate
description
Move Machine
Items
ID
name
base_price
description
purchase
amount
final_price
discount
date
City
Area
cardinality
connect
Route
length
owns
use
usable
Catch
Teach
Battle
loser
winner
ID
XP
gold
start_time
duration
winner
loser
Fight
life_remaining
damage_inflicted
damage_recieved
Belongs to
Evil criminal
money_steal
signature_phrase
leader?
Sidekick
Pokémon trainer
ID
name
experience
gold
Team
belong
Nature
ID
like_flavour
dislike_flavour
stat_increase
stat_decrease
Has
Region
Operate in
Criminal organization
name
base_location
Owns
found
Pokémon creature
ID
is_baby
nickname
level
experience
effort
HP
attack
defense
Sp. Attack
Sp. Defense
speed
evasion
remaining_health
status_condition
position
method
time
date
Cause/Recieve
Belongs to
Species
ID
name
weight
height
basic_experience
has
Encounter
ID
meeting method
Find
condition_type
condition_value
probability
Evolve
Gender
Time
Level
Base Statistics
ID
effort
HP
attack
defense
Sp. Attack
Sp. Defense
speed
evasion
Move
name
accuracy_percentage
PPs
power
priority_order
target_type
damage_class
Type
name
Affect
A2B (multipler)
B2A (multiplier)
Ability
name
short_description
long_description
hidden
Level
number
necessary xp
Growth rate
title
formula
Healing
life_points
Damage
strength
Status
chance_to_apply
special_effect

## 2.2 Pokémons module justification




This first module of the project focuses on defining Pokémon creatures and their characteristics. To start off, it is important to distinguish between a ***Pokémon Creature*** (instance of a specific specimen) and a ***Pokémon Species***, which encompasses a set of Pokémons. This distinction is one of the first things we spotted and decided to create one entity for each of these, specifying that a *Pokémon Creature* belongs to a single *Species,* while a *Species* can have a different set of creatures. Hence, we connected these two entities with a 1 to N relationship.

As mentioned in the statement, all creatures within the same species share some characteristics: *ID, name, weight, height,* and *basic experience* (base points gained by each species when their team wins a battle), which we defined as attributes of the *Species* entity, so that all the specimens from the same species could access them. We know their identifier will be unique to each species (to store their information in the Pokédex), so we decided to make this its primary key.




Additionally, each creature from the same species shares some basic statistics (*health points, attack, defense, special attack, special defense, speed, and evasion*), with an *effort multiplier* to determine their stats' development. As these are innate to each species, we decided to create an entity ***Base Statistics*** that holds all of the base values for the mentioned fields, and directly relate it with *Species* (with a 1 to 1 relationship, as a species will only have one set of base statistics, which is unique to them). Therefore, whenever a

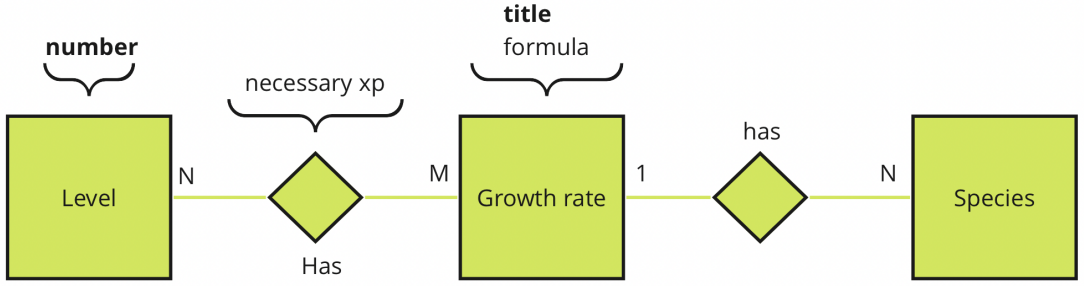
new Pokémon creature is created / levels up, it will inherit these base values onto its own respective attributes, allowing the system to update them throughout the battles, while maintaining their base value for new specimens. Considering different sets of base statistics could have repeated fields, we decided to create an attribute *ID* to clearly identify these (making it the entity's primary key).

Moreover, a Pokémon also has a series of ***Abilities*** which are innate to each species, which is why we connected these two entities with an N to M relationship, as a species can have up to three abilities, and the same ability can be shared within different species. We considered there wouldn't be any abilities with the same *name*, so we defined this as its primary key, while also storing both a *short and an extensive description* of its effect. We also had to contemplate the possibility of a species having a hidden ability. To enable this functionality, we had to know for each species and each of their abilities if it was hidden or not, so we decided to store an attribute *hidden* in the relationship between these two entities. We had to define this as a shared attribute as we took into account the possibility of a particular ability being hidden for a specific species, but not for the others. On the contrary, if we had assumed that when an ability is hidden no species can use it, we could have just added this attribute to the *Ability* entity.

Furthermore, Pokémon *Species* can be of a certain ***Type***, an entity that defines their resistance to other Pokémon types and the efficiency of their moves. Each type has a unique *name* (Water, Fire, Dragon…), which we defined as its primary key. We connected this entity with *Species* in an N to M relationship, considering a Pokémon species can have a secondary type, and there can be different species with the same type.

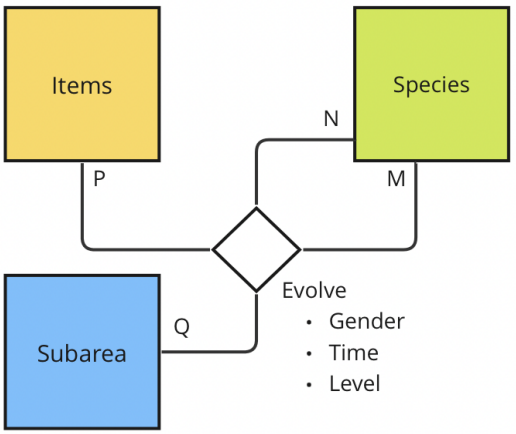
To determine the effects of a certain Pokémon type to another, we created a reflexive relationship "***affect"*** in this entity with a cardinality of N to M (as each type affects the others in a different way, and each type is affected by others differently). As attributes of this relationship we defined two *damage multipliers* (one from type A to B and from type B to A). These damage multipliers will define the way each type affects the others, so they must be shared attributes (as for instance Type Water will affect different Fire than Dragon). If each type always affected the rest in the same way, we could have defined the damage multiplier as an attribute of the *Type* entity.

We decided to add two damage multipliers to avoid having "mirrored relationships" (from A to B and from B to A). By adding this second attribute, we can get all the information we need from a single relationship, avoiding the use of unnecessary memory and making the information more compact and easily accessible.



This section also defines the process of a Pokémon leveling up, which depends on each species' ***Growth Rate,*** which can be of different types depending on its title ("Slow", "Fast"…). We decided to create this entity and add a title as its primary key (as no growth rates can share the same title), and connect it to the *Species* entity with a 1 to N relationship (as a species can only have one growth rate, but this one can be shared within different species). For each growth rate, we also want to know its *formula* (attribute that will determine the speed at which a Pokémon levels up).

To know whether a Pokémon has acquired the *necessary experience points* to reach the next level, these will be an attribute of the relationship between the entities *Grow Rate* and ***Level***, as we considered different Pokémon species might require a different number of experience points to reach the same level. If we had thought all species required the same points, we could have added this as an attribute to the *Level* entity, which only holds its primary key: the *number* of the level it corresponds to (as we won't have different levels with the same number). The cardinality between these two entities is N to M, as a growth rate defines the experience points necessary to reach all levels (from 1 to 100), and a level can eventually be reached with any growth rate.

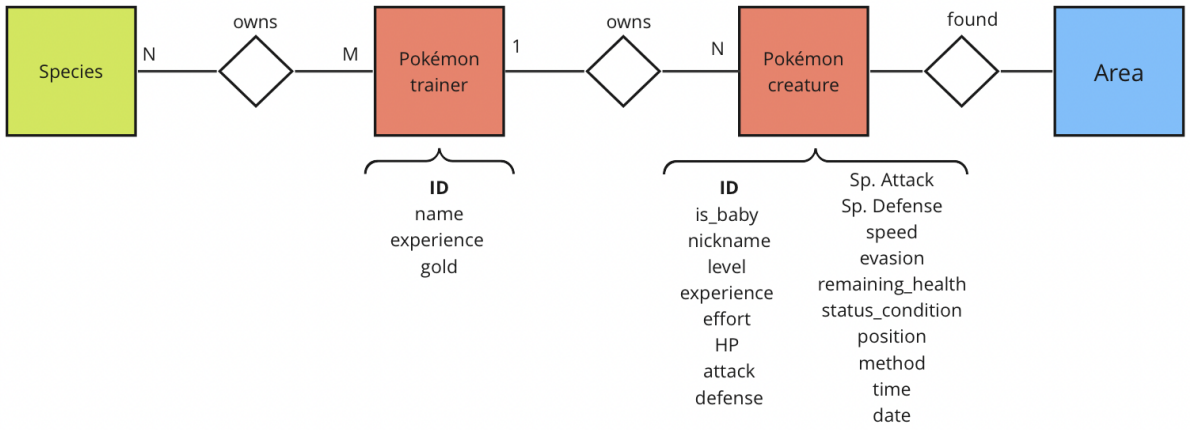



When a Pokémon reaches a specific level, it can evolve into other species. Hence, we defined the reflexive relationship "***evolve"*** between different Pokémon *Species.* We decided that the most appropriate cardinality for this relationship would be N to M, considering a Pokémon can evolve into different species, and a specific species can be reached from evolving different species. Apart from reaching a specific *level,* some other requirements need to be met for a Pokémon to evolve. Thus, we also store as attributes of this relationship the following: *gender, time, location,* and the use of a specific *object.* All of these fields may or may not be required for a specific species to evolve, so we will allow them to be NULL if they are not necessary for the evolution. Also note that the *location* and *object* requirements are connections to their respective entities (which will be explained in further sections).

From the above-mentioned entities, *Pokémon Creature, Species,* and *Type* have connections to other modules, which will be specified in their corresponding sections.

## 2.3 Trainers module justification

The players of the system are introduced in this module together with the battles and their logistics, which we will now define.

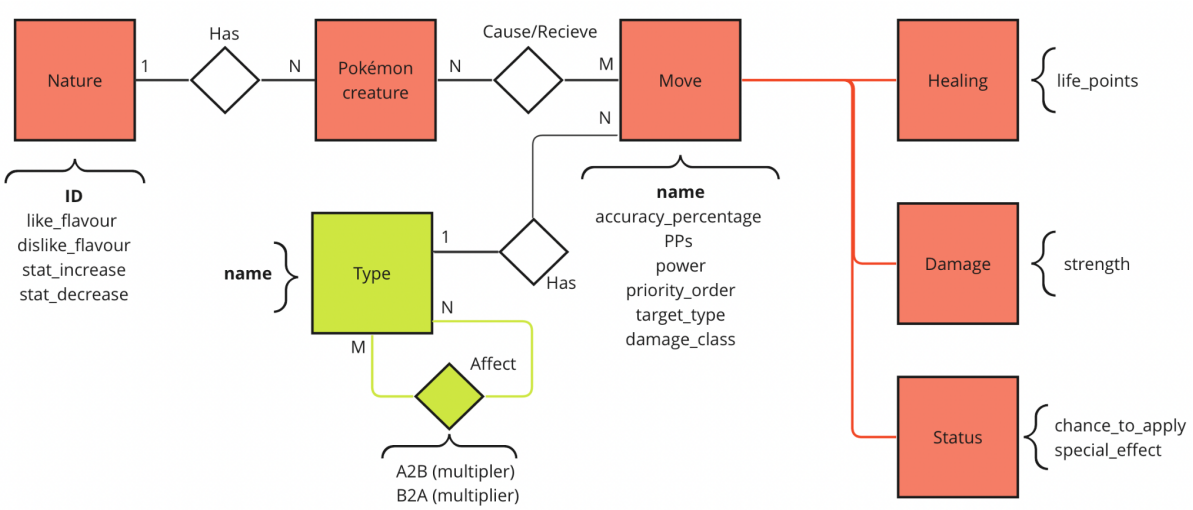



Each player is referred to as a Pokémon ***Trainer*** and has a series of attributes stored in the entity we created, such as: *name, experience points,* and *amount of gold.* As none of these are necessarily unique to a player, we decided to add an *identifier* as its primary key.

Each trainer owns a series of ***Pokémon Creatures*** and for each of them we want to know the *method* used to capture it, the *time* of the encounter, the *date*, and the *location.* So we had to create a 1:N relationship between *Trainer* and *Pokémon Creature* (as a trainer can have an infinite amount of Pokémons, but a Pokémon can only be owned by one trainer) and store all of these attributes in the *Pokémon Creature* entity. However, the *location* attribute eventually became a 1:N relation from the *Pokémon Creature* entity to the *Area* entity (detailed explanation in section 2.5), as a Pokémon Creature can only be encountered in one area, but several creatures could be encountered in the same area.

Besides owning *Pokémon Creatures,* trainers also store all the information about the ***Pokémon Species*** they own in their Pokédex, so we created a N to M relationship between these two entities to keep track of the species the trainer has and represent this collection of information (considering the trainer can have several species, and a single species can be owned by different trainers).

For each *Pokémon Creature* we also want to know their *nickname, level,* and *experience,* so we also stored all of these attributes in its entity. We believed the nickname of a creature wouldn't be unique, as explained on section 2.3, these creatures are instances of a Pokémon Species, specimens that could be duplicated. Therefore, we decided to add an *identifier* and make it the primary key of this entity.

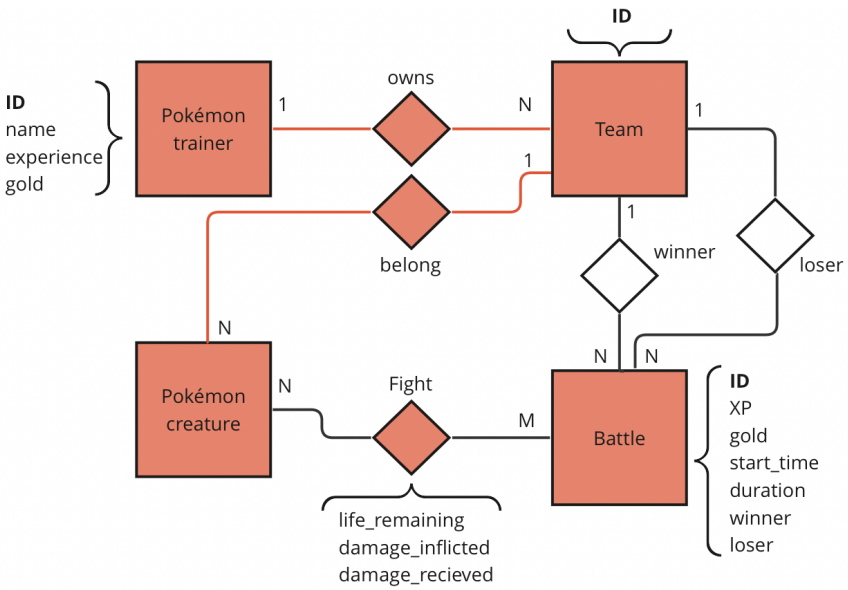



In addition, *Pokemon Creatures* also have a unique ***Nature***, a set of characteristics that affects their statistics and tastes. We defined this as an entity with the following attributes: *flavor they like, flavor they dislike, stat to increase, stat to decrease* (these stats will always be modified a 5%). Once more, as none of these attributes seem to be unique, we defined an *identifier* and made it the entity's primary key. We connected these two entities with a N:1 relationship, considering a creature can only have one nature (as it is unique), but the same nature can be shared by different creatures.

The game also defines ***Moves***, which are the actions taken by the *Pokémon Creatures* during a battle. We connected these two entities with an N to M relationship (as a Pokémon can have up to 4 moves at a time, and the same move can be applied by different Pokémons). Each of them has a *name, accuracy percentage, number of power points* (times it can be used), *power, priority order, target type* (anyone, another Pokémon or itself), and *damage class.* From these, we considered the *name* to be unique (as it wouldn't make sense to have two moves called in the same way), so we made this its primary key.

Moreover, we need to know the move's *Type*, so we connected these two entities with a 1 to N relationship, as a move can only have one type, but there can be several moves from the same type.

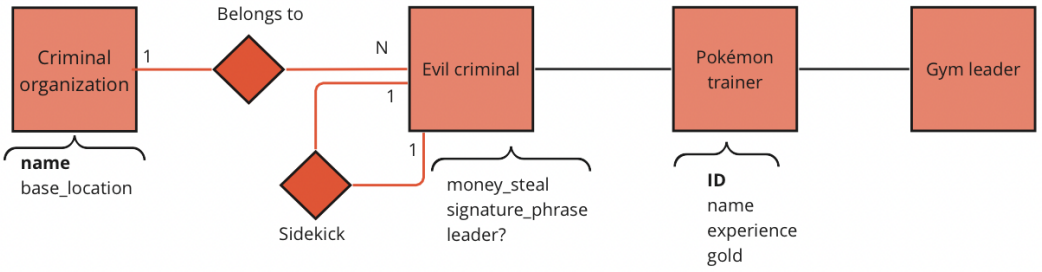
As specified in the statement, there can be three types of moves: ***Healing***, from which we want to know the *life points* it recovers; ***Damage***, from which we want to know the *strength;* and ***Status***, which defines a *special effect* to be inflicted into another Pokémon and a *change to apply* such effect. We created these three entities and stored the necessary information as attributes. However, as these were all moves, we realized we would need to store all the predefined attributes in these three entities. Instead, we decided to make *Move* a super-entity that generalized these three sub-entities, which will inherit all the attributes from it (including its primary key, so they don't need one) and also have their personal ones. By doing so, we would remove repetitive content and simplify our diagram.




To take part in a battle, *Trainers* organize their *Pokémon Creatures* in ***Teams.*** Hence, we defined a 1:N relationship between *Trainer* and *Teams* (as we assumed a trainer can have more than one team, but a team can only belong to one trainer), and a 1:N relationship between *Team* and *Pokémon Creature* (as a team has from one to six creatures and a creature can only belong to one team at a time). We added an *identifier* as its primary key, as there were no other attributes that needed to be stored in this entity.

During a *Battle*, the *Trainer* must know the *position* of the creature in the team, their *remaining health,* and their *status condition* (all added as attributes to a *Pokémon Creature*). Moreover, for each *Battle* and *Pokémon Creature*, the trainer must know their *remaining life,* the *damage inflicted* on them, and the *damage received.* These attributes cannot be stored in the *Pokémon Creature* entity like the above-mentioned, as they must be accessible after a battle ends (not only during its course). Meaning that, if we stored them in the *Pokémon Creature* entity, these would keep changing every time they took part in a new battle, rewriting their previous values. Therefore, we must store them as shared attributes between the *Battle* and *Pokémon Creature* entities, which we will relate with an N to M relationship (as a Pokémon can participate in several battles and several creatures fight in the same battle).

These teams then fight in 1vs1 ***Battles***, which award a number of *experience points* and offer a *gold reward* (both attributes from the *Battle* entity). From these battles, we also want to know their *start time* and *duration.* As none of these are strictly unique to a *Battle*, we created an *identifier* and made it its primary key. To determine the *winner* and the *loser* of each *Battle*, we created two 1:N relationships that connected a *Team* and a *Battle* (as a battle only has one winner and one loser, but a team can participate in different battles). We considered storing this information in a single N:M relationship (two teams fight in the same battle and a team can participate in different battles) with a shared attribute *winner* that would be false if the team lost and true if the team won. However, we considered this approach more intuitive in terms of the future queries that will have to be made into our database to extract the desired information.




Furthermore, there can be two types of trainers: ***Gym Leaders*** and ***Evil Criminals*****.** We created these two sub-entities and made them inherit the attributes from *Trainer* (now a super-entity), including its primary key. *Gym Leaders* are those who can be battled in their own *Gym* (entity explained in section 2.5).

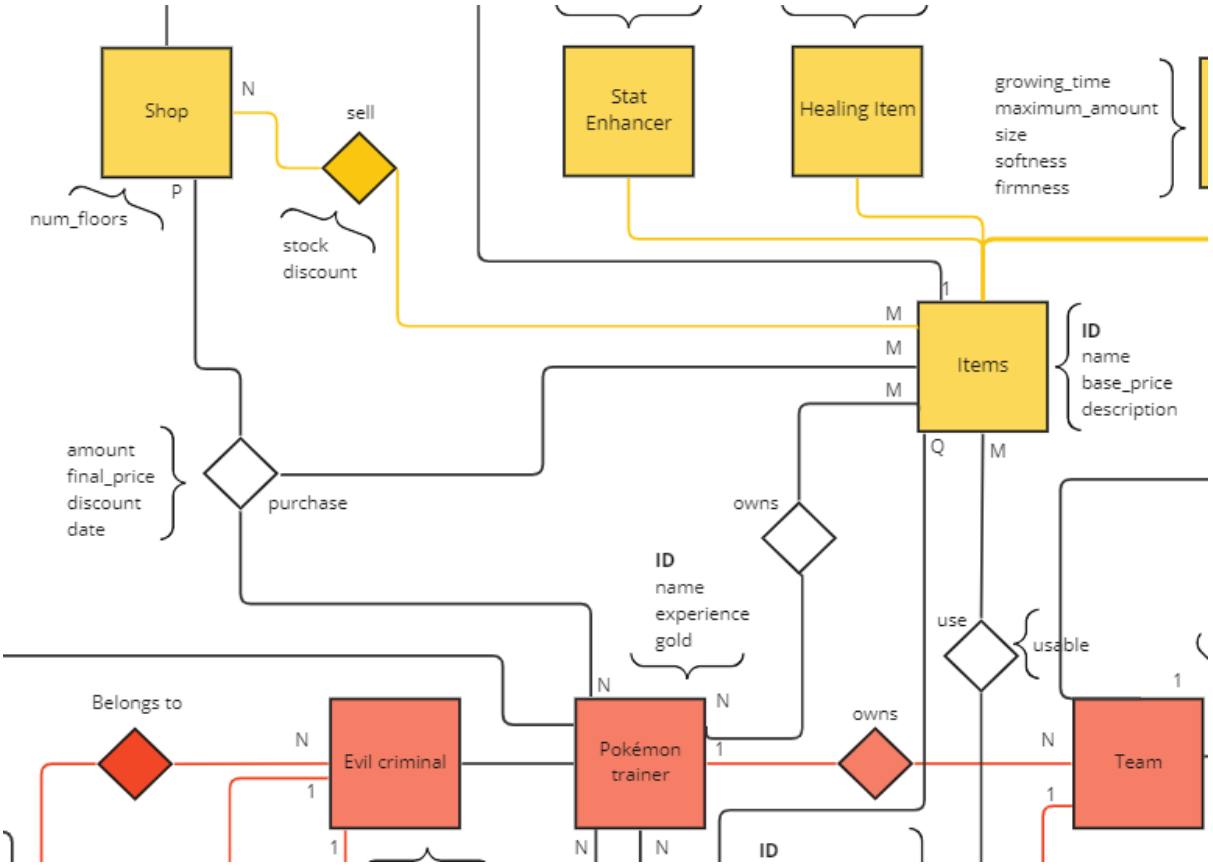
On the other hand, *Evil Criminals* are those that *steal money* when they win another player, and say their *signature phrase* before starting a battle (both added as attributes). Each of them has an "***evil sidekick"*** (another *Evil Criminal* they are always with), so we added a reflexive 1:1 relationship to this entity to define this companionship (1:1 as they can only have one sidekick and they cannot be the sidekick of more than one *Evil Criminal*).

These trainers belong to a ***Criminal Organization*** which has a *name, base location,* and a *region.* We added all of these as attributes to the *Criminal Organization* entity that connects with a 1:N relationship to *Evil Criminal* (as an *Evil Criminal* belongs to only one organization, but an organization is made out of several criminals). Also, one of them will be the *leader* of this organization, so we added this as an attribute to the *Evil Criminal* entity (making it false if they are not and true if they are). If we considered they could belong to more than one organization, we would have to put this attribute in their relationship (as a criminal could be the leader of an organization but not of the others). We considered the *name* of the *Criminal Organization* to be unique, so we made it its primary key. Additionally, we had to connect this entity to *Region* (explained in section 2.5) to indicate the *region* attribute specified before.

## 2.4 Shopping module justification

In this module, the first thing that we had to consider was whether we were going to create an entity backpack for the trainers or not. We ended up deciding that we did not need it since we could simply have two different relationships between ***Items*** and ***Trainers***. Firstly, we have an N to M binary relationship between the *Items* and the *Trainers* that indicates that they own these items. The other relationship we have is a ternary relationship between the *Trainers*, the ***Shops*** and the *Items*.

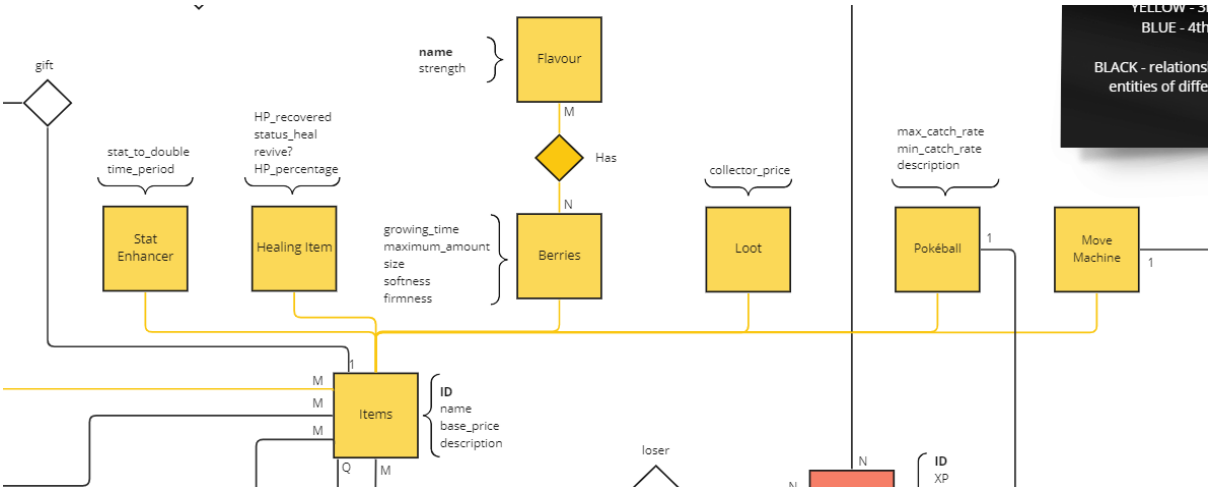
We separated the relationships in these two since *Trainers* related to the *Items* in two distinct ways. On the one hand, they own the *Items* and the amount of each item that each trainer has is described in their N to M relationship. If we wanted to know how much of an item a trainer has we would make a JOIN between trainer and the item name with the COUNT functionality. We preferred this approach rather than including an attribute in the relationship which we could have called "amount". On the other hand, trainers will buy items from the shops in an N, M, P relationship called purchase. This relationship includes a series of attributes called: *amount, final_price, discount* and *date.* These are stored in the relationship and will function as a record to save the specific information about each purchase.




As for the attributes of each entity, *Items* have three attributes including their *name* (primary key since we considered that there won't be two *Items* with different names), their *base_price* and the *description*. The *Shop* has an *ID* (primary key), their *name* and the number of floors (*num_floors*). There is one final N to M relationship between the *Shops* and the *Items* named as "sell" which also contains two attributes, *stock* and *discount*. It should be

noted that this discount is different from the previously mentioned one that appears in the ternary relationship since that one is the *discount* that the items had at the moment the purchase was made which can differ entirely from the *discount* currently assigned by the *Shop*.

In order to simplify the diagram, *Items* has been defined as a super-entity with six sub-entiies, all of which are different types of *Items* and share the same attributes as their parent entity. First of all are the ***Stat_enhancers*** which will double a specific statistic of a Pokémon (*stat_to_double*) and the *time_period* that the enhancement will last for. Secondly, there are ***Healing items***. We considered whether we wanted to make another division for each different type of healing items but we concluded that it would be easier to implement if we differentiate them through their attributes. That is why these specific items have: i) *HP_recovered* which indicates how many health points will the item cure, ii) *status_heal* which names the special effect that the items cures, if any (the attribute will be null in case that it does not have this effect), iii) a boolean named *revive* and iv) *HP_percentage* which are only used if the *Healing item* can revive the Pokémon and have it regain consciousness with a specific percentage of its own HP stat.




Another type of *Item* that we defined were the ***Berries.*** Each of these have a *growing_time*, a *maximum_amount* that can grow on a tree at the same time, a *size*, a *softness* and a *firmness*. Each of these *Berries* will have one or more ***Flavours*** (which is why their relationship is N to M since a Berry can have more than one flavour but each flavour may be present in more than one Berry). This entity *Flavour* has a *name* (primary key since there won't be more than one flavour with the same name) and a *strength*. Fourthly, there are ***Loot*** items which are only there for collection and monetary purposes. This is the reason why their only attribute is their *collector_price* a particular price that is different form the regular *base_price* present in every type of *Item* since a *Trainer* could sell them for an increased value. The "collector" is a simple NPC and does not constitute a different type of *Trainer.*

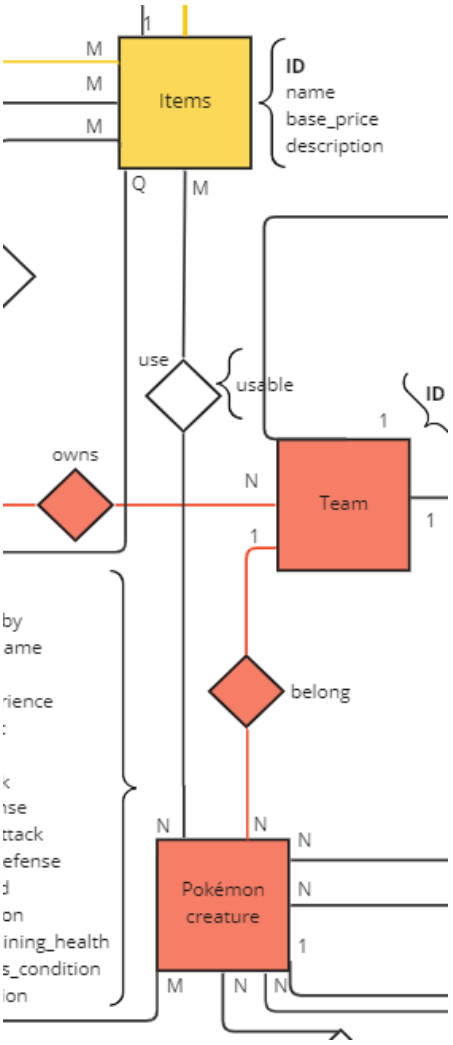
Furthermore, another very important *Item* to consider are the ***Pokeballs***. These have a *minimum_catch_rate* and a *maximum_catch_rate* which indicate how good or bad the *Pokeball* is compared to others and a *description* which indicates the conditions to be met to use the *Pokeball*'s maximum capacities. Naturally, these are used to catch *Pokemon creatures* which is why there is a 1 to 1 relationship between those entities. Each *Pokeball*

may only be used once to catch one instance of a *Pokemon creature*. Should it fail, the *Pokeball* is forever lost but if it succeeds then that creature will live and be stored in that *Pokeball* forever, without the possibility of changing to another one.

Finally, the last type of *Item* are the ***Move_machines*** which are used to teach a specific *Move* to an instance of a *Pokemon creature*. Each *Move_machine* will only be able to teach one *Move*, therefore, their relationship is 1 to 1.

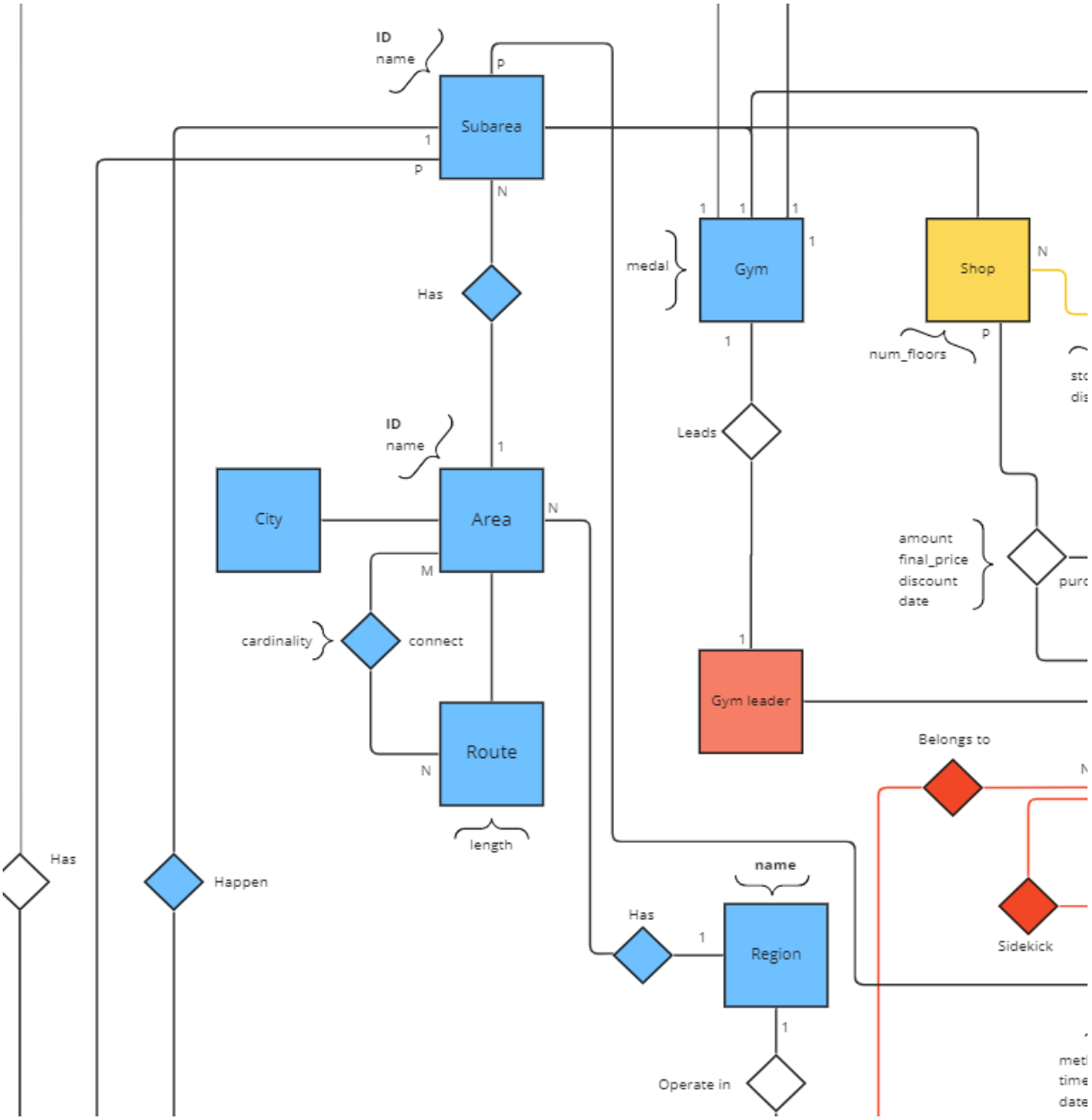
It should be noted that some of these *Items* may be required for Pokémon to evolve.

The last piece of information to be explained in this section is that the *Pokémon creatures* have a special N to M relationship with the entity *Items* called “use”. Each creature will be able to use any of the previously described *Items* and any of them could be used in any and many creature. There are some particular *Items* that can be used in battle which is indicated as a boolean in the relationship between these and the *Pokemon creatures.* We considered the possibility of including this as an attribute to the *Items* themselves but it made more sense to add it in the relationship since this is something that should be taken into account when *Pokémon creatures* engage in *Battle.*



## 2.5 Exploration module justification

The fourth and final module that is included in this project is the exploration section. The in-game world has several ***Regions*** each with their own particular *name* (primary key). These *Regions* are composed of ***Areas*****,** each with their own *ID* and *name* (the ID attribute was included since we have considered that some *Areas* could share names with others), and each *Area* is at the same time composed of some ***Subareas*** which also have an *ID* and *name* (the same reason can be applied to this ID too).




As previously stated, each *Region* shares a 1 to N relationship with the *Criminal organization* entity since more than one organization can work in the same region but those organizations cannot belong to more than one *Region*.

The *Area* entity is a super-entity to two sub-entities, the ***Cities*** and the ***Routes*** (these have two attributes called *ID* and *length*). The *Routes* in particular have another relationship with

its parent indicated as the way there connect to other areas. This is an N to M relationship because more than one *Route* can connect an *Area* with others but at the same time an *Area* will have more than one connection thanks to the *Routes*. There is an attribute in this relationship called *cardinality*. This will either be north, east, west or south depending on whether the *Route* connects the *Area* through one of those cardinalities. For a time we considered this whole way that *Areas* relate to *Routes* as reflexive relationship which we wanted to call route as well. However, we realized that it was a mistake since tha wouldn't have allowed us to indicate the way in which the *Areas* were connected through their cardinal points.

Similarly, *Subarea* is a super-entity with the previously mentioned ***Gym*** and *Shop* entities. The *Gyms* have a *Type* associated to them which means that all the *Pokémon creatures* of the *Gym Leader* assigned to it will share the same *Type*. In order to indicate this we have created a 1 to 1 relationship between the entities *Gym* and *Type*. The *Gyms* also have a 1 to 1 relationship with *Items* since whenever a trainer defeats a *Gym* they will be gifted one particular item (which will always be the same). Finally, and most importantly, *Battles* will take place in these *Gyms* in which a trainer will challenge other trainers and the *Gym Leader* in order to be able to gain the valued *medal* and the gift item. It is worth mentioning that *Battles* can occur in places other than the *Gyms* but the only moment in which we are interested in knowing where it was is in the case of *Gyms.* Another clarification is that the medal does not constitute an *Item*, since it cannot be used nor sold.

The *Subareas* will have ***Encounters*** that can happen in their perimetres. the *Encounter* entity is the one that will allow the trainers to catch a new Pokémon for their collection in the first place. The relationship is 1 to N as a *Subarea* may have more than one *Encounter* but each encounter will only happen in one particular *Subarea*. These *Encounters* will have an *ID* (primary key) and a *meeting_method* which will show how the encounter can happen. In addition, and in order to indicate which *Species* of Pokémon may appear in these *Encounters* that the players may have, we have included an N to M relationship between *Encounter* and *Species.* There is a possibility that many *Species* appear on any *Encounter* which is why the relationship is N to M and also because, we understood that we needed to add some attributes to this relationship like: the *condition_type* and *condition_value* that need to be met in order for the *Encounter* for that *Species* to occur and the *probability* that it will happen even if the conditions are met.

# 3 Relational model

Once we had already completed the conceptual model previously explained and thoroughly revised it, we proceeded to design the relational model, the intermediate step in between the first diagram and the final implementation of the database.

This one consisted of translating the entities of the conceptual model into tables with a set of rows (one for each attribute) and columns, in which we would specify if the respective attribute is a primary key (PK) or a foreign key (FK). The foreign keys would be those formerly specified which cannot have repeated values and cannot be null. On the other hand, foreign keys are those that refer to the primary key of another table. We will use these last ones to represent the relationship between different tables.

The link between different entities will be translated differently depending on its cardinality. For instance, a 1:1 relationship will only consist in adding a FK to one of the tables involved (we will have to decide to which one it makes more sense to be added in terms of design). Moreover, the 1:N relationship between two entities will be represented in the same way, only that in this case the FK will be added to the table with the N cardinality (as each of the instances of that table will only have 1 of the others it refers to). On the contrary, an N:M relationship requires the creation of a new table that connects the ones involved. This one will have both as PK and FK the PKs of the entities it connects. Therefore, the creation of this new table will allow us to add shared attributes to it.

## 3.1 Diagram

In the following page you will find a screenshot of our relational model diagram. If you have any troubles reading it, please access the following link:

https://drive.google.com/file/d/1WE8djb_wTsuU7IR-2d2A3Fun4BukowFA/view?usp=sharing

**Shop**

| | |
|---|---|
| PK / FK | id_shop |
| | num_floors |

**sell**

| | |
|---|---|
| PK / FK | id_shop |
| PK / FK | name_item |
| | stock |
| | discount |

**Flavour**

| | |
|---|---|
| PK / FK | name_flavour |
| | strength |

**Has**

| | |
|---|---|
| PK / FK | name_flavour |
| PK / FK | id_berrie |

**purchase**

| | |
|---|---|
| PK / FK | id_shop |
| PK / FK | name_item |
| PK / FK | id_trainer |
| | amount |
| | final_price |
| | discount |
| | date |

**Stat Enhancer**

| | |
|---|---|
| PK / FK | id_stat_enhancer |
| | stat_to_double |
| | time_period |

**Healing Item**

| | |
|---|---|
| PK / FK | id_healling_item |
| | HP_recovered |
| | status_heal |
| | revive? |
| | HP_percentage |

**Berrie**

| | |
|---|---|
| PK / FK | id_berrie |
| | growing_time |
| | maximum_amount |
| | size |
| | softness |
| | firmness |

**Loot**

| | |
|---|---|
| PK / FK | id_loot |
| | collector_price |

**Pokéball**

| | |
|---|---|
| PK / FK | id_pokeball |
| | max_catch_rate |
| | min_catch_rate |
| | description |
| FK | id_creature |

**Move Machine**

| | |
|---|---|
| PK / FK | id_move_machine |
| FK | name_move |

**Gym**

| | |
|---|---|
| PK / FK | id_gym |
| | medal |
| FK | name_item |
| FK | name_type |

**SubArea**

| | |
|---|---|
| PK | id_subarea |
| | name |
| FK | id_area |

**Gym Leader**

| | |
|---|---|
| PK / FK | id_gym_leader |
| FK | Gym |

**Evil Criminal**

| | |
|---|---|
| PK / FK | id_criminal |
| | money_steal |
| | signature_phrase |
| | leader? |
| FK | sidekick |
| FK | Criminal Organization |

**owns_items**

| | |
|---|---|
| PK / FK | name_item |
| PK / FK | id_trainer |

**Item**

| | |
|---|---|
| PK / FK | id_item |
| | base_price |
| | description |
| | name_item |

**uses**

| | |
|---|---|
| PK / FK | name_item |
| PK / FK | id_creature |
| | usable |

**Criminal Organization**

| | |
|---|---|
| PK | name_organization |
| | base_location |
| FK | name_region |

**Pokémon Trainer**

| | |
|---|---|
| PK | id_trainer |
| | name |
| | experience |
| | gold |

**Team**

| | |
|---|---|
| PK | id_team |
| FK | Pokémon Trainer |

**Battle**

| | |
|---|---|
| PK | id_battle |
| | XP |
| | gold |
| | start_time |
| | duration |
| FK | Team_winner |
| FK | Team_loser |
| FK | Gym |

**fight**

| | |
|---|---|
| PK / FK | id_battle |
| PK / FK | id_creature |
| | life_remaining |
| | damage_inflicted |
| | damage_recieved |

**Region**

| | |
|---|---|
| PK | name_region |
| | name |

**has_species**

| | |
|---|---|
| PK / FK | id_trainer |
| PK / FK | id_specie |

**Pokémon Creature**

| | |
|---|---|
| PK | id_creature |
| | is_baby |
| | nickname |
| | level |
| | experience |
| | effort |
| | HP |
| | attack |
| | defense |
| | sp. Attack |
| | sp. Defense |
| | speed |
| | evasion |
| | remaining_health |
| | status_condition |
| | position |
| | method |
| | time |
| | date |
| FK | Area |
| FK | Trainer |
| FK | Team |
| FK | Species |
| FK | Nature |

**Nature**

| | |
|---|---|
| PK | id_nature |
| | like_flavour |
| | dislike_flavour |
| | stat_increase |
| | stat_decrease |

**City**

| | |
|---|---|
| PK / FK | id_city |

**Area**

| | |
|---|---|
| PK | ID |
| | name_area |
| FK | name_region |

**evolve**

| | |
|---|---|
| PK / FK | id_current_specie |
| PK / FK | id_next_specie |
| | gender |
| | time |
| | level |
| FK | id_item |
| FK | id_subarea |

**cause_or_receive**

| | |
|---|---|
| PK / FK | id_creature |
| PK / FK | name_move |
| PK | id |

**connect**

| | |
|---|---|
| PK / FK | id_area |
| PK / FK | id_route |
| | cardinality |

**Route**

| | |
|---|---|
| PK / FK | id_route |
| | length |

**Species**

| | |
|---|---|
| PK | id_specie |
| | name |
| | weight |
| | height |
| | basic_experience |
| FK | id_statistics |
| FK | Growth Rate |

**Base Statistics**

| | |
|---|---|
| PK | id_statistics |
| | effort |
| | HP |
| | attack |
| | defense |
| | Sp. Attack |
| | Sp. Defense |
| | speed |
| | evasion |

**Move**

| | |
|---|---|
| PK | name_move |
| | accuracy_percentage |
| | PPs |
| | power |
| | priority_order |
| | target_type |
| | damage_class |
| FK | Type |

**Encounter**

| | |
|---|---|
| PK | ID |
| | meeting_method |
| FK | id_subarea |

**find**

| | |
|---|---|
| PK / FK | id_encounter |
| PK / FK | id_specie |
| | condition_type |
| | condition_value |
| | probability |

**has_ability**

| | |
|---|---|
| PK / FK | id_specie |
| PK / FK | name_ability |
| | hidden |

**has_type**

| | |
|---|---|
| PK / FK | id_specie |
| PK / FK | name_type |

**Type**

| | |
|---|---|
| PK | name_type |

**Healing**

| | |
|---|---|
| PK / FK | id_healing |
| | life_points |

**Damage**

| | |
|---|---|
| PK / FK | id_damage |
| | strength |

**affect**

| | |
|---|---|
| PK / FK | name_type1 |
| PK / FK | name_type2 |
| | type1_to_type2_multiplier |
| | type2_to_type1_multiplier |

**Status**

| | |
|---|---|
| PK / FK | id_status |
| | chance_to_apply |
| | special_effect |

**Growth Rate**

| | |
|---|---|
| PK | title_growth_rate |
| | formula |

**Ability**

| | |
|---|---|
| PK | name_ability |
| | short_description |
| | long_description |

**has_level**

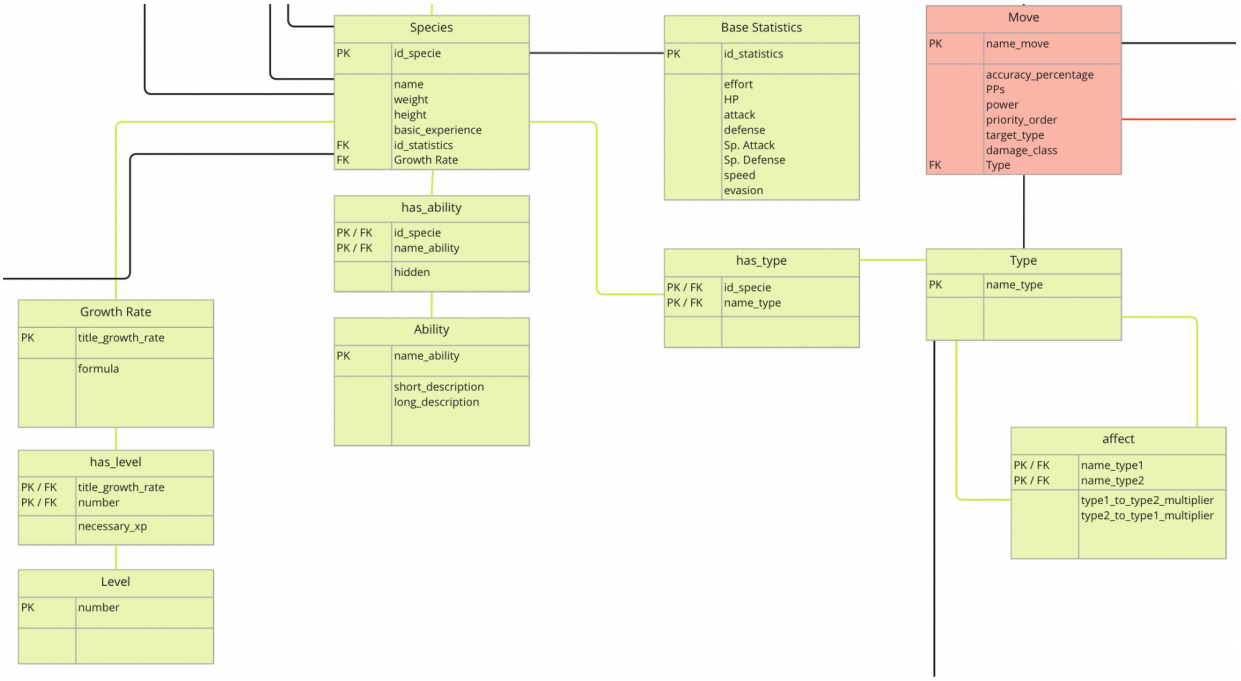
| | |
|---|---|
| PK / FK | title_growth_rate |
| PK / FK | number |
| | necessary_xp |

**Level**

| | |
|---|---|
| PK | number |

## 3.2 Pokémons module justification

As previously described on section 2.2, this module connects the entities ***Pokémon Creature, Pokémon Species, Base Statistics, Ability, Growth Rate, Level,*** and ***Type***. We first created all these tables and copied their PKs and attributes from the conceptual model. Then, we proceeded to translate their relationships.




As specified before, a ***Pokémon Creature*** has only one ***Pokémon Species***, but a Species has several Creatures. As this is a 1:N relationship, we just added the species of each creature as an attribute to the *Pokémon Creature* table with a FK to the table *Species.*

Regarding a **Pokémon Species' *Base Statistics***, as this was defined as a 1:1 relationship, we had to decide to which table we should add a FK to the other. We concluded it was better to add it to the *Pokémon Species* table, in case we had to consider the possibility of two species having the same Base Statistics, in which scenario the relationship would become 1:N (as the species would still only have one set of base statistics), but the attribute wouldn't have to be replaced.

The relationship between ***Ability*** and ***Pokémon Species*** is N:M, meaning we were required to create an extra table that connects these two. We added the primary keys of both entities as the PK and FK of this new connecting table (has_ability). It was not necessary for us to extend the composite PK by adding an extra attribute as we did not contemplate the possibility of a species having the same ability more than once, as we considered it didn't make a lot of sense in the game. However, we did add the shared attribute *hidden* in this new table so that it referred to the connection between both entities.

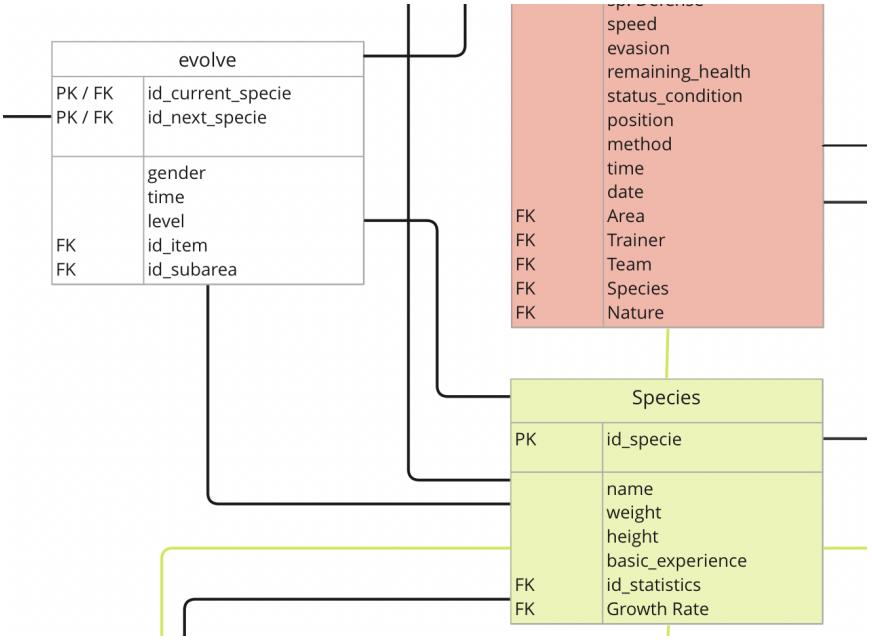
Concerning a ***Pokémon Species' Growth Rate***, we added a FK that pointed to the PK of the *Growth Rate* table to the *Species* table. We took this approach as this was defined as a

1:N relationship. By adding this attribute, we are allowing each species to have a different growth rate (without any restrictions about repeating growth rates between species).

Furthermore, each ***Growth Rate*** has a ***Level*** it refers to and we must store the necessary experience points to reach it. Therefore, as this is a N:M relationship, we created the table *has_level* which has a composite key made of both of the table's PKs (while also storing them as FKs to connect both tables). In this table we also added the *necessary experience points* as an attribute, considering these would change depending both on the level to reach and the growth rate each species has. As there can't be two ways of reaching the same level with the same growth rate (repeated relationship), it wasn't necessary for us to add an extra attribute to the composite key.

In addition, we also created an extra table for the relationship between ***Pokémon Species*** and ***Type***, as this one had an N:M cardinality. We added both as a PK and a FK the primary keys of its respective connecting tables. In this case, as we did not consider the possibility of a species having the same type more than once (which we believed didn't make sense), we didn't have to extend the composite key to allow repetitions.

We created a table for the N:M reflexive relationship in the *Type* entity. This one ("***affects***") has as a PK and as a FK the name of the types it connects (PK of the *Type* entity) and holds the two previously mentioned damage multipliers as attributes. In this case, we believed a type couldn't affect another in different ways (for instance Water VS Fire will always hold the same damage multipliers). Therefore, we didn't add any attribute to the composite key to allow for repetitions.

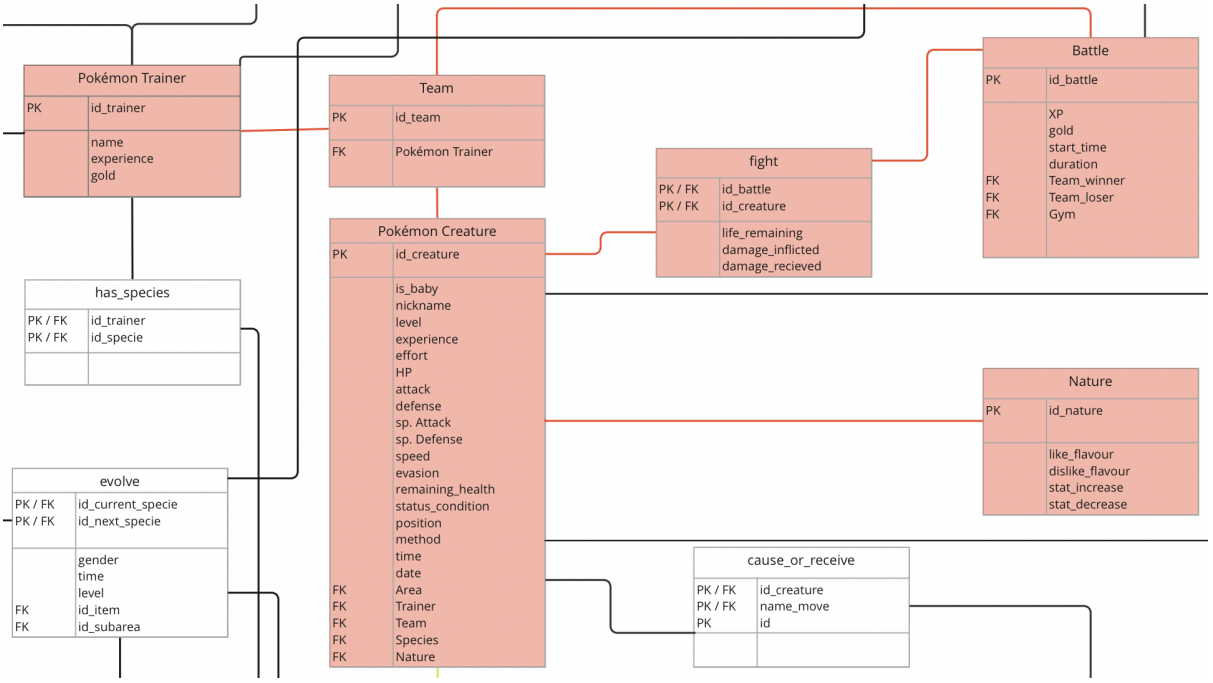



Concerning the "***evolve***" reflexive relationship in the *Pokémon Species* entity, we had to create a new table (as the relationship has a cardinality of N:M and stores some attributes) which has as both PK and FK the identifiers of the two species it connects (the one that evolves and the one it evolves to). As attributes of this relationship, we previously defined the *level* required to evolve, the *gender*, the *time*, the *location,* and the use of a specific *item*. All of these were added as attributes to this new table. However, the *location* and *item*

attributes are FK to the tables *Subarea* and *Item*, respectively (as these hold all the information we need to access). At first we considered the possibility of the same Species evolving multiple times (repeated relationships). Therefore, we had to extend the composite key of this *evolve* table by adding an *identifier* attribute (making it also its PK) to allow the same species to evolve more than once. However, we realized that this relationship only holds the requirements to evolve, it doesn't actually store any information about the particular evolutions (*time* when it happened, or *location*). In this case, there will be no repetitions and we won't make use of the extended composite key, so we decided to remove it.

## 3.3 Trainers module justification

As previously described on section 2.2, this module connects the entities ***Trainer, Pokémon Creature, Nature, Move, Healing, Damage, Status, Teams, Battle, Evil Criminal, Gym Leader,*** and ***Criminal Organization***. We first created all these tables and copied their PKs and attributes from the conceptual model. Then, we proceeded to translate their relationships.



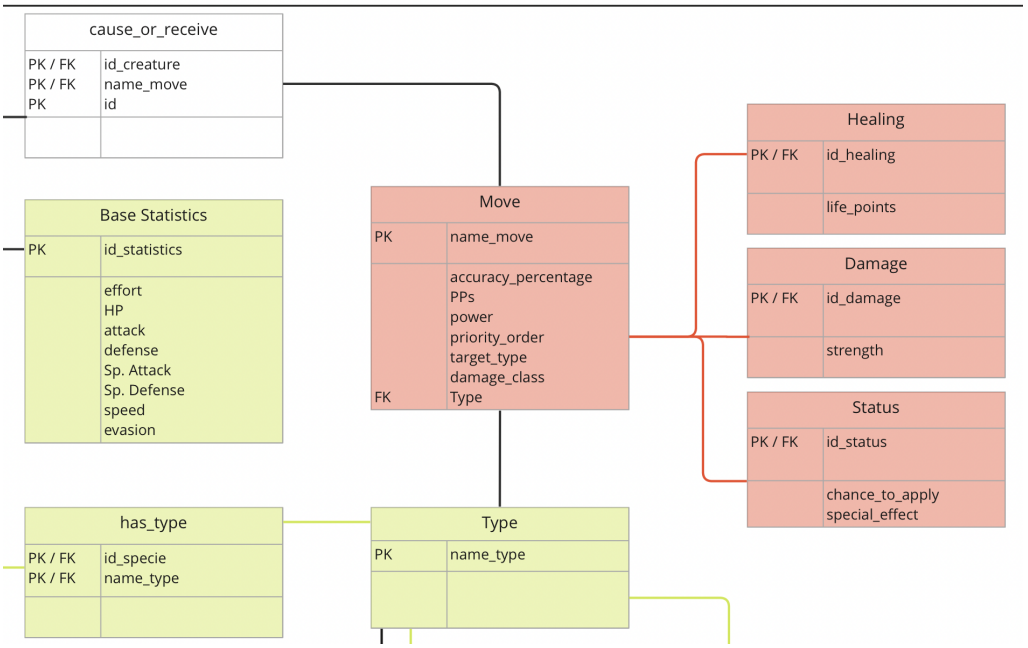

As previously specified, a ***Trainer*** owns a series of ***Teams*** (1:N relationship as a team can only be owned by one trainer). Therefore, to translate this relationship we just added an attribute *trainer* to the *Team* table (FK to the *Trainer* table). Similarly, a ***Trainer*** also owns a set of ***Pokémon Creatures*** (also 1:N relationship), so repeated the same process and added an attribute *trainer* to the *Pokémon Creature* table (FK to the trainer's ID). Moreover, as for each creature we wanted to know the ***Area*** where it was captured, we transformed the 1:N relationship as an extra attribute to the *Pokémon Creature* table (FK to the *Area* table). In addition, to know the ***Pokémon Creatures*** that form each ***Team,*** we added a FK attribute to the *Pokémon Creatures* table to specify to which team they belong (knowing they can only be part of one team at a time).

The ***Pokémon Creatures*** in a team can fight in ***Battles,*** so we created a table that related these two entities (*fight*) and added as a PK and as a FK the Pks of both. We took this approach as a *Pokémon Creature* can fight in different *Battles* and several creatures fight in the same *Battle.* However, as a *Pokémon Creature* cannot fight several times in the same battle (wouldn't make sense), there is no need for us to extend the composite key of the *fight* table. Moreover, we included the different attributes that we want to store for each creature fighting in a battle in this relationship.

Out of the two ***Teams*** involved in a ***Battle***, there is always a *winner* and a *loser*. To store this information, we added two attributes that are FKs to the *Team* entity (*team_winner, team_loser*), as a battle will only have one winner and one loser, even though a team can be the winner/loser of multiple battles (1:N relationship).

Besides, a ***Trainer*** also has a set of ***Pokémon Species,*** an N:M relationship that was translated by creating a new table *has_species* in which we store as Pk and as FK the primary key's of both entities. As a trainer wouldn't own a species twice (as the species doesn't refer to the specific creature, it just represents whether a trainer owns any creature from a certain species) we did not add any extra attributes to the composite key of this table.

***Pokémon Creatures*** also have a unique ***Nature***, which can be shared between different creatures. Again, we translated this 1:N relationship by adding an attribute to the *Pokémon Creature* table called *nature*, which is a FK to the *Nature's* PK.

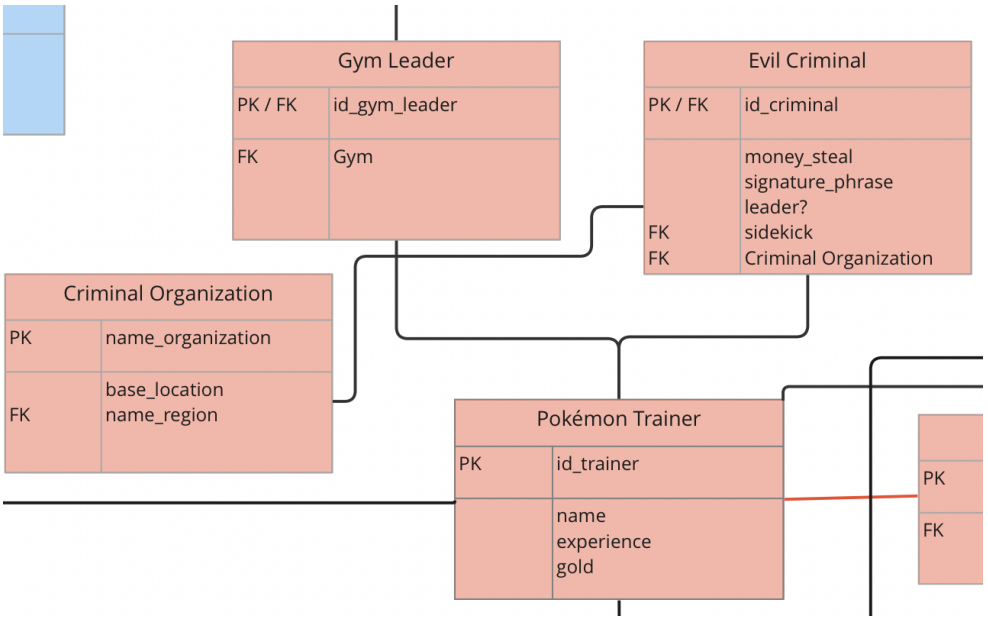



Moreover, ***Pokémon Creatures*** can also have ***Moves.*** However, this relationship is different as we must allow them to have more than one move (N:M relationship). Hence, we created a table *cause/receive* that related these two entities. This one has as its composite key the PKs of both entities (while also being FKs). However, as we wanted to allow a *Pokémon Creature* to use the same move more than once (in different battles for example), we added an *identifier* to its composite key to ensure we allowed repetitions and to distinguish between the different instances of the table.

Each ***Move*** has a specific ***Type,*** so we added an attribute to the *Move* table that specified which type it belonged to. We followed this approach as it describes a 1:N relationship (a move can have only one type but there could be different moves with the same type).

To represent the generalization of the ***Move*** table to the ***Healing, Damage,*** and ***Status*** entities, we added the primary key of the super-entity *Move* as both the PK and FK of the

sub-entities. We took this approach as it wouldn't make sense to have a Healing move without a Move entity, so all of them relate to an actual Move and wouldn't exist without it.



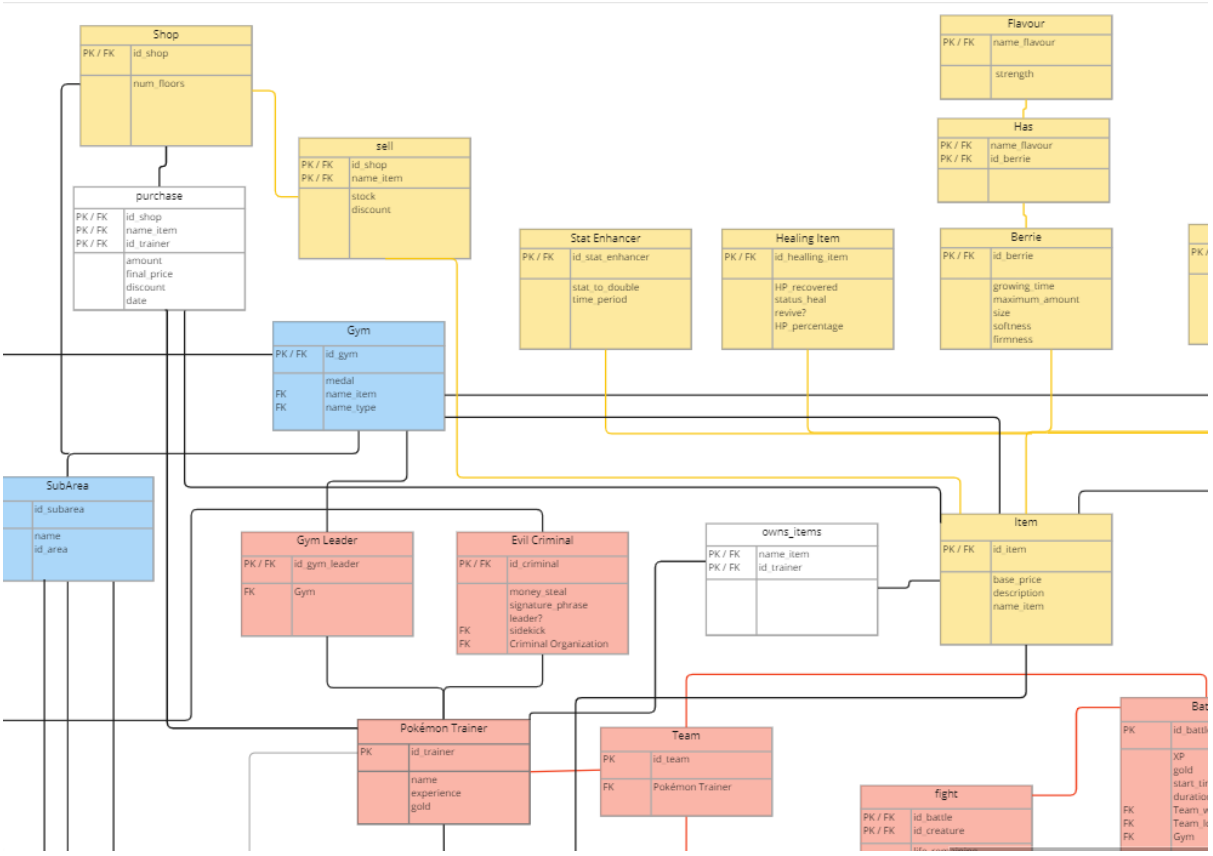

Furthermore, the system must allow two types of ***Trainers: Gym Leaders*** and ***Evil Criminals***. Once more, we used a generalization to represent this relation. Therefore, we added the primary key of the *Trainer* table to the sub-entities *Gym Leader* and *Evil Criminal* as both a PK and FK. Regarding the *Evil Criminals,* we represented its reflexive relationship (*evil sidekick*) as an attribute in its table, as this was only a 1:1 relationship.

Lastly, we also know that ***Evil Criminals*** belong to a ***Criminal Organization***. To represent this 1:N relationship we added an attribute to the *Evil Criminal* table which is a FK to the *Criminal Organizations'* PK (as we considered an *Evil Criminal* can only belong to one *Criminal Organization*, even though an organization is made out of several criminals).

## 3.4 Shopping module justification

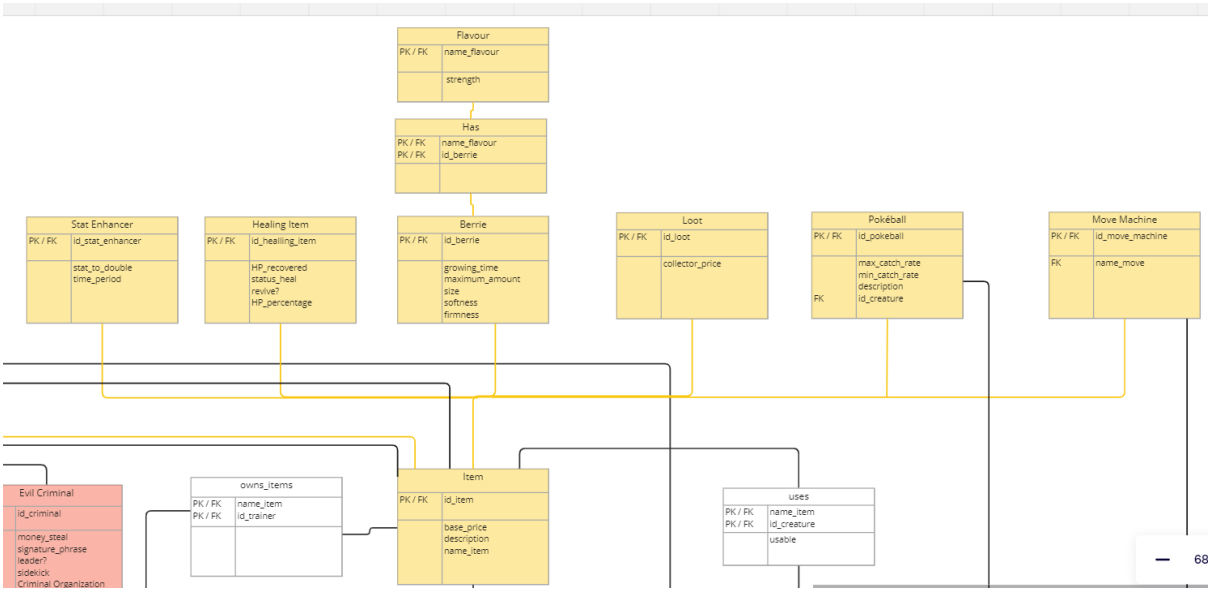
Firstly, since we had an N to M binary relationship between the ***Items*** and the ***Trainers*** that indicated that they own the items, we created a new table between the two entities called *Owns* which includes the *name of the item* and the *id of the trainer* as both FK and PK.

When it came to the other ternary relationship between ***Shop***, *Items* and *Trainers,* we had to implement a new table called *purchase* which has the *id of the shop, name of the item* and *id of the trainer* as its PK which are also FK which come from their respective connecting tables. All the attributes that were previously in the relationship as far as the conceptual model was concerned were included in this new table now.




Additionally, we have created another new table for the N:M relationship between *Shop* and *Items* called *Sells.* It again has the PK/FK components that come from the other tables' PK and has the attributes that used to be in the relationship. This approach allows for several *Items* to be sold in more than one *Shop.*

To represent the generalization of the ***Item*** table to the all of the six sub-entities, we added the primary key of the super-entity *Item* as both the PK and FK of the sub-entities. Naturally, it would not make sense to have, for example, a Stat Enhancer Item without an Item entity.




Since the *Berries* had the particualrity of having ***Flavours*** (N:M), we had to include a table *Has* in order to relate which *Flavours* were present in each *Berry.*

The 1:1 relationship between *Pokeball* and *Pokemon creatures* was translated in the form of adding the *id of the creature* as a FK in the *Pokeball* entity. This would help to understand which *Pokemon creature* was the *Pokeball* used on. Similarly, the attribute *name_move* was added in the *Move machine* table to refer to which *Move* could the item teach.

Lastly, since the *Pokemon creatures* need to be able to *Use* these *Items,* we created a new table with the *name of the item* that was used and the *id of the creature* that used it as the PK/FK of the table. It also includes the boolean that indicates whether the item is *usable* in battle or not.

## 3.5 Exploration module justification

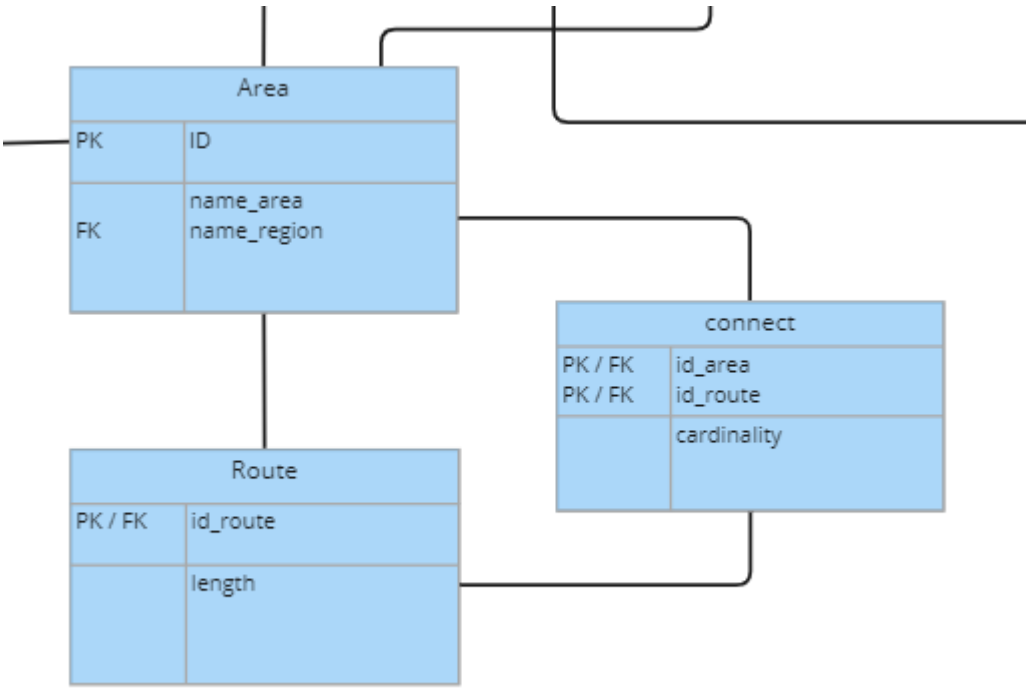
The game has to allow for several ***Regions*** to exist in the same world. Each of these may have one or more *Criminal Organizations* working in that territory. This has been accomplished by adding a FK attribute in the Criminal Organization entity which signals in which Region they work at.

Each *Region* is composed of several ***Areas***. Therefore, since this used to be an 1:N relationship, we added the *Region ID,* as a FK attribute of each of the *Areas.* Likewise, we had to do the same to be able to show to which Area did each ***Subarea*** belong to.

As for the generalizations, we took the same approach we used for the other generalizations previously mentioned. The ***Cities*** and the ***Routes*** have the super-entity *Area*'s ID as the PK of those entities which are at the same time a FK. The same situation happens with the ***Shops*** and ***Gyms*** that share the same ID as their super-entity *Subarea.*

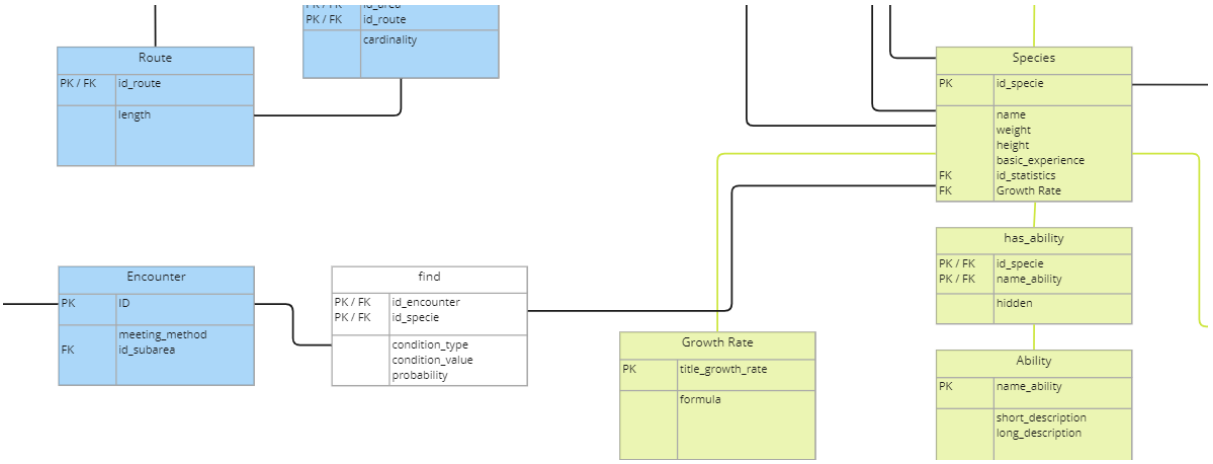
It is worth noting that the *Gym* entity has two extra FKs as attributes, one *item name* and one *type name* since we have to be able to know which item does the gym give whenever a trainer defeats it and also which type are all the Pokemon in the *Gym* characterised by.

*Areas* are connected to other Areas though *Routes* which are themselves Areas. As this is understandably an N:M relationship, we have created a new table called *connect* that relates to both Areas and Routes through its PKs which are also FKs, the *id of the area* and *id of the route*.




Finally, there are ***Encounters*** that can happen in the *Subareas* and are related to them through a FK attribute named *id_area*. This entity had an N:M relationship with the *Pokémon Species* that was translated into a new table called *Find*. This table has two PK/FK, one for

each PK of the tables it connects and has the attributes that used to correspond to the ones present in the relationship in the conceptual model.



# 4 Conclusions

## 4.1 Use of resources

| Stage | Valeria | Eugènia | Sebastián | Marc | Total |
|---|---|---|---|---|---|
| **ER model** | 14 | | | | 14 |
| **Relational model** | 3 | 3 | 0 | 0 | 6 |
| **Total:** | 17 | 17 | 14 | 14 | 20 |

As we previously mentioned, all of us worked in the ER model together for about 13 hours: 5 hours individually reading the statement and doing our first proposal of the database, and 9 hours going through those with the whole team. We believed this was the best approach to ensure everyone in the group was familiar with all the modules and to allow constructive discussions that would eventually lead us to the best solution (four heads are better than one). This is why the total amount of hours is equivalent to each of the team's work hours.

In terms of the relational model, you can see how Valeria and Eugènia did all the work. This is because Sebastián and Marc wrote this report while they did the diagram. We considered that having reached the point where the conceptual model was done, we could split the remaining work to save some time.

Overall, all the team members spent more or less the same amount of hours working on the project.

## 4.2 IA use

We did not use any AI tools to develop this project.

## 4.3 Lessons learnt and conclusions

This first phase of the project has helped us reinforce all the concepts previously acquired throughout the first semester of the subject and has shown us how to apply the theory learned to a real-life example, which has been a really exciting and enriching experience.

We have also learned the importance of good group communication and cohesion and how this can help in the designing process. The constructive discussions we had with each other made us all try to see things from the others' perspective, which was a very valuable and necessary experience, as it made us realize that our initial thought might not be the best one and we were able to interpret the statement in a way we couldn't have done by ourselves. Hence, we managed to improve and optimize our drafts until we came up with the best possible version of the diagrams we could have designed.