# Final Design Document

# Team Trojans

## Mission Statement:

**The purpose of the Pokémon Day Care Database is to manage information about trainers' Pokémon, and their stays at day care.**

Objectives:

The objective of the database (DB) is to make it easy for the day care owners to keep track of all the tedious data for the trainers and Pokémon, to make it easy and efficient to contact trainers about their Pokémon's updates. The database (DB) will also move the daycare to a paperless way of storing data in the cloud for a seamless experience. It will automatically generate and send updates to the trainers. The database will keep track of critical data for Pokémon development and training. This will make it easy for a layman to log and retrieve a Pokémon's activity history and health data at the daycare. This will also help us perform some data analytics to understand the Pokémon's performance, evolution progress and their social life. In addition, DB will be able to get an overview of the business's performance by analyzing the friendships created, number of new trainers, numbers of trainers lost, Pokémon evolution etc.

Executive Summaries

The entities being tracked in the Pokémon daycare databases are:

1. Trainers
   DB will be storing essential information about trainers such as their address, phone number, email. This information allows us to communicate with the trainers effectively and provide them with the best experience. Moreover, as DB send frequent updates being able to access their email and phone number enables us to do that effectively. In addition, DB needs this information to contact them in case of an emergency.
2. Pokémon
   Pokémon go hand in hand with trainers. Since trainers are linked to a Pokémon, DB will be keeping track of immense information for each Pokémon. DB will be keeping track of their name, their species (Pikachu, Charizard, Bulbasaur), who is their owner, their level, and their gender. Moreover, DB also store information on the Pokémon's progress. For instance, if they increase their level, DB will send updates to their trainer and update the database with their new level. The database also allows us to know what kind moves the Pokémon has learnt as well, or the number of eggs that they have

conceived with their friends. The DB allows us to know the relationship of one Pokémon with others and if they are close enough, they conceive eggs with another Pokémon.

3. Events:

   To keep track of the Pokémon activity DB will record the events that take place in their lifetime. For instance, once they are conceived, DB keep track on every life event. If they grow to learn a move, it will be set in stone in the db. If they start developing friendships, and from those friendship comes if they conceive an egg that event will be recorded as well. It is essential to have these events being tracked since DB want to provide weekly updates to trainers on the new developments that took place. With events the DB also keep track of the current level of the Pokémon, along with the moves history, eggs and friendships with other Pokémon. We have 3 main types of events: level ups, new moves learned, and eggs conceived. These are all tracked through the events table.

4. Species:

   We also keep track of the species (Raichu, Dragonite, etc) of Pokémon and each specie is associated with a certain type (grass, poison or water) so DB knows the type/types for each Pokémon specie.

5. Moves:

   We will keep track of all the moves that the pokemon's have learned and allow other pokemon's to learn new moves from within all the moves learned by previous pokemons. This way we will know what moves the day care is best in nurturing into Pokémon of a specific specie and use that to our advantage.

6. Eggs:

   We keep track of all the eggs that the pokemon have conceived along with their parents. The date conceived of the egg is recorded, we keep track of the egg's name as well.

7. Specie Types:

   This keeps track of all the different types of pokemon that exist. E.g. Rock, water, poison and etc.

All tables in the database:

Entity tables:

1. Pokémon: This tables tracks all details that pertain to a pokemon (name, gender, level) along with their trainer id that we can use to trace back to their trainer. There is also an event ID which is used to trace back all the events that took place for that Pokémon. Moreover, there is a species id which is used to figure out the Pokémon's specie

2. Trainers: This table as expected relates the trainer id in the pokemon table to the trainer's information which is stored in this table, i. e number, email and address.

3. Events: This table tracks all the events that take place in a Pokémon's lifetime, it informs us of the date, event type (level up, new move or egg). In addition, it also has a pokemon id which is used to trace the event back to the Pokémon.

4. Moves: It keeps track of all the moves that any of the pokemon in the daycare has learned.

5. Species: This table has the list of all the species that exist in the world of pokemon, any pokemon which is entered in the daycare has this species associated with it so that the day care can cater to its needs accordingly.

6. Types: This is the different type of Pokémon nature or specie types. Each specie is associated with certain number of types and the day care needs this info so that it can put it in the associative environment. Grass Pokémon will be put in the grassland and water pokemon will be put into water.

7. Eggs: This table keeps tracks of the egg id and the egg name. Egg id allows us to trace the egg back to its parents.

Non-entity tables:

1. Egg events: This table links the event id to the egg id so the egg can be associated to a father and a mother

2. Pokémon Friendships: This table establishes a two-way relationship between Pokémon that decide to become friends in the day care.

3. Species types: This table links a Pokémon specie (Bulbasaur) with its associated specie types (grass and poison)

4. Move events: This table links the new move that a pokemon has learned to its event id and move id, so it is easy to identify the new move that the pokemon has learned.

5. Level up events: This keeps track of the level changes for a pokemon, it has associated event id which link the change in level to a certain event id which is transitive relation related to a pokemon. Thus, the level change is related to a pokemon

6. Event type: This table just saves the name of all the different types of events that can take place. I.e., Level up, new move or egg.

All Tables and their details

## Pokemon Table

This table holds information about pokemons kept at the daycare center. This is the main table in this database. If any of the tables in this database needs to refer to information about pokemon, it refers to this table using pokemon_id. When a new pokemon comes to the daycare center, that pokemon is added as a new entry to this table.

- pokemon_id INT AUTO_INCREMENT PRIMARY KEY
- pokemon_name VARCHAR(50)
- species_id INT
- trainer_id INT
- pokemon_level INT
- is_female BOOLEAN

- **pokemon_id** is the primary key which uniquely identifies each pokemon entry. We use unique integers that are auto-incremented in every insertion. The decision to use auto incremented integers as this table's primary key was made because other candidate keys or any combinations of them do not satisfy the requirements to be the primary key.
- **pokemon_name** is the field that holds the nickname of each pokemon. The name is represented as the char value type and can hold up to 50 characters.
- **species_id** is a foregin key that references the species table's species_id table. The field uses the species table as a validation table to make sure only valid pokemon species are entered into the field. Restrict is applied as a deletion rule to this foreign constraint. If a species in the species table is referenced in the pokemon table, a deletion operation on it will be rejected.
- **trainer_id** is also a foreign key, and it references the trainer table. It is used to store the reference to the trainer that owns the pokemon. Restrict deletion rule applies to the foreign constraint, preventing the deletion of a trainer record from the trainer table if the

trainer is referenced in the pokemon table. Up to two pokemons can refer to the same trainer (i.e. a trainer cannot check in more than two pokemons)

- **pokemon_level** is a field that holds an integer value, representing the current level of the pokemon. The field has the unsigned constraint to make sure that pokemon level is always positive (we assumed that pokemon level cannot go negative).
- **is_female** is the field with the boolean value type. If TRUE, the pokemon is male, If FALSE, it is female.

## Trainer Table

This table holds information about trainers. A trainer checks in up to two pokemons at the daycare. Each pokemon belongs to one trainer. The Trainer table is referenced by the pokemon table.

- tainer_id INT AUTO_INCREMENT PRIMARY KEY
- phone_number VARCHAR(50)
- email VARCHAR(255)
- address VARCHAR(50)

- **trainer_id** is the primary key of this table which uniquely identifies each trainer. The choice to use an id field as a primary key is made because any other candidates keys or any combinations of them do not satisfy the requirements for a primary key.
- **phone_number** is the field with the data type of char. The phone number will be stored without any special characters. The decision to allow up to 50 characters is made to accommodate trainers who have a foreign phone number (more than 10 digits).
- **email** stores the email address of the trainer. Although the maximum length of characters in an email address is more than 255 characters, the decision was made to only allow registration of emails that have at most 255 characters.
- **address** stores the address of the trainer. This address field only allows up to 50 characters. The decision was made that 50 characters is enough to express most of the address types around the world. We intentionally did not create validation tables for the address field. Nor did we separate the address field into individual address components. Since the customer of the daycare center is rather global, we avoided assuming any specific address format. The business domain and business specifications do not require the collection of the geographical data of the trainers. As such, the priority of this field is not to collect accurate locational data. This field exists for only individual lookup purposes. We don't intend any aggregate data to be produced based on the location of trainers.

## Species Table

Species table is used as a validation table for the species field of the pokemon table. We decided to create this validation table as we want to make sure that the entry to the species field of the pokemon table is a valid species name. One species can be referenced by multiple pokemons. A pokemon can reference only a single species (we assumed there are no hybrid pokemons).
The attention of the correctness of the species field of the pokemon table is justified by the business's possible use of pokemon's species information to produce aggregate data, such as the performance of pokemons of a specific species at the daycare. If a specific species is not

doing well (not leveling up as often as other species, or learning fewer new moves), the business might benefit from hiring employees who specialize in training of that specific species.

- species_id INT AUTO_INCREMENT PRIMARY KEY
- species_name VARCHAR(50)

- **specied_id** functions as the primary key of the table. The decision to use the id field, instead of name field, as the primary key is made as we probably use a join operation on pokemon and species tables anyway. We expect species information to be presented along with the information on species kind, which requires a joint operation.

## Species Types Table

This is the linking table that represents the many to many relationship between pokemon species and pokemon types (one pokemon species can have different pokemon types depending on the life stage of the pokemon). Both fields, species_id and type_name, are foreign keys and serve as the primary key.

- species_id INT
- species_type VARCHAR(50)

species_type has a foreign key constraint and has a deletion rule of RESTRICT (i.e. if a type is referenced by this table, it cannot be deleted from the Types table).

## Types Table

This table functions as the validation table for the type_name field of the species table. This field stores all possible pokemon types, such as Electric or Ice. This field can store up to 50 characters as we assume there are no pokemon species names that have more than 50 characters.

- species_type VARCHAR(50)

## Events Table

This table is the gateway for the events data stored in the database. This table indexes all events (i.e. giving event ids to all events. This does not mean “indexing” for efficient lookups). It also stores the date and time when the events take place. This table also stores event type as we need to know which table to look up when querying further information about that particular event (i.e. The table name specified in the FROM clause in a join operation will be taken from the event_type field). One pokemon can have multiple events. However, an event belongs to only one pokemon.

- event_id INT
- pokemon_id INT

- date
- event_type VARCHAR(50)

**event_type** is a foreign key and it references the event_type field of the Event Tybe table. The deletion rule of RESTRICT is applied to the foreign key constraint on this field.

## Event Type Table

This table stores the name of the three event types and serves as a validation table for the event_type field of the events table. The event_type field of this table stores event type names using characters. The table only contains three rows (egg_event, level_up_event, and move_event). These event type names match the name of each corresponding event table. This is because event_type is used when joining the Event table with corresponding event tables, such as the Egg Event table and Level Up table). An event is only identified as belonging to a single event type.

- event_type VARCHAR(50)

## New Moves Table

This table exists to store move names for the new move events. This table is a child table of both the Level Up Event table and the Event Table. This table refers to the Level Up Event table to make sure that only one move event exists per level up event. The New Moves Table also refers to the Event table as new move events need their own date, which is stored in the Event table. Whenever pokemons learn new moves, the new move event is logged to this table, along with the name of the move that is learned. The name of the move is validated using the Moves table. An entry to the New Moves table refers to a single event in the Event table and to a single level up event in the Level Up table.

- event_id INT
- move_name VARCHAR(50)

**move_name** uses the Moves table as its validation table. The foreign key constraint on this field has the deletion rule of RESTRICT. If the name of the move is referenced by any of the entries in the New Moves table, it cannot be deleted from the Moves table.

## Moves Table

This is a validation table for the move_name field of the New Moves table. This table has only one field, move_name. This field stores all possible moves that pokemons can learn. This field can store up to 50 characters. We deemed all move names to be less than 50 characters.

- move_name VARCHAR(50)

## Level Up Events Table

Entry to this table will be triggered when the pokemon_level field of the Pokemons table is updated (entry to the Event table will be also triggered for level up events). This table is used to store the history of all level up events. Whenever the pokemon_level field of the Pokemon table is changed, that change will be recorded in this table. This table stores the new level reached as an integer. We don't store previous levels as we assumed that pokemon only acquire one level at a time. If the increment amount is the same for every level up, we don't need to store the previous level as we can calculate its value.

- event_id INT
- level_reached INT

## Egg Events Table

This table records egg events. When pokemon conceives an egg, that information goes into this table. As both male and female pokemons contribute to the fertilization, an egg event belongs to two pokemons. To avoid duplicate information about the egg, however, the name of the egg (pokemons name their eggs) will be stored in another table (Egg Table). Some information about the egg events are, however, duplicated. For example, as the event belongs to both male and female pokemons, the date of the event will be duplicated in the Events table (nevertheless, because each pokemon might recall different date time for when exactly fornication took place or which fornication activity lead to the fertilization, we deemed having two dates for the same event rational).

- event_id INT
- egg_id INT

## Egg Table

This table stores the information about eggs produced at the daycare. This stores information about names of eggs. This table is referenced by the Egg Events table. We assumed that we don't look up eggs parents. That's why we don't store information about eggs' parents in this table. Rather, through tables related to events, each pokemon can trace the eggs that they produced. When the business wants to know how they performed in the sense of the number of eggs produced, they look up the number of total rows in this table, as it is the total number of eggs produced at the daycare center.

- egg_name VARCHAR(50)

## Friendship Table

This table stores information about friendships made at the daycare center. This table stores two pokemon ids stored in the Pokemon table. A friendship in this table is intentionally duplicated in two entries. If you have a friendship between Pokemon A and Pokemon B, one entry stores Pokemon A in the first pokemon id column and Pokemon B in the other. In the other entry, Pokemon B is stored in the first column and Pokemon A in the other. This is to make sure that any lookup for friendship does not fail. A deletion rule is applied on the foreign constraint for

the pokemon ids. If a pokemon is referred to in the friendship table, that pokemon cannot be deleted from the Pokemon table.

- pokemon_id_1 INT
- pokemon_id_2 INT

## Business rules and their justification

1. Trainers can only have 2 Pokémon check in at once.

This means that there cannot be more than 2 Pokémon associated with one trainer's id at once in the Database, we will have a view which keeps count of the number of Pokémon currently in the DB linked to a specific trainer id. If a trainer has less than two Pokémon then they can check in another pokemon otherwise they will be prompted to check out one of their pokemons to check in a new one or not check in the new pokemon at all. This business rule was a required expectation from the client as they did not want to cater to more than two pokemons from one trainer

2. Pokemons cannot learn a move they have already learnt

Each Pokémon's moves learnt are recorded in the events table and within the events table are linked to the move events table. From these move events we will check if the new move that the Pokémon has learnt is it a move that already exist in the DB, if it exists then it means that the pokemon is basically relearning a move it has already learnt and it does not seem reasonable. Therefore, if the new move is a move that already exist in the DB for that Pokémon then the new move will be not learnt as the pokemon already learnt the move in a previous event. This would stop a pokemon from learning the same move again and again as we want to provide updates to the trainers and not repeat the same thing again and again

3. Two trainers cannot have the same phone number

Since weekly updates are provided to trainers for their pokemons in the day care, we cannot have two numbers which are the same for two different trainers as each Pokémon is associated with a different trainer. For this we will check the phone numbers that exist in the DB for all trainers and if the phone number already exist then it cannot be used, the uniqueness of the phone number will be applied which will allow the day care to reach out to different trainers for their specific pokemon.

4. Eggs should be linked with a female and male pokemon

Since only a female and a male can conceive eggs, the egg that is harvested to be linked to parents of different genders. For this we will be keeping track of the father_id which is an alias for the pokemon_id. This father_id has to be associated with a male pokemon meaning the is_female field in the pokemon table should be false and the same goes for mother_id it's Boolean should be true. This would constrict the egg to be related to two different genders.

5. The date for a new event must be after the last event, and not null.

A new event being generated must have a date that is not null and must be after the last event that was added in the DB. For this we will do a field check when it is being entered and check it with the last event that was entered in the DB. If the date for the last event is greater than the new event then the user will be asked to enter a new date that meets the requirements. Moreover, the date will be defaulted to the current timestamp.

6. Level cannot go down.

This will be a constraint on the levels field in the level up table, this will constrict the types of arithmetic operations that can be performed on the field. Since a level of a Pokémon cannot go down, we will implement in the DB the fact that we check the last level up event of the pokemon and check their level in that last event, the new level should be greater than the last level in the last level up event. This will not allow pokemon levels to go down as that does not happen in pokemons and they can only level up.

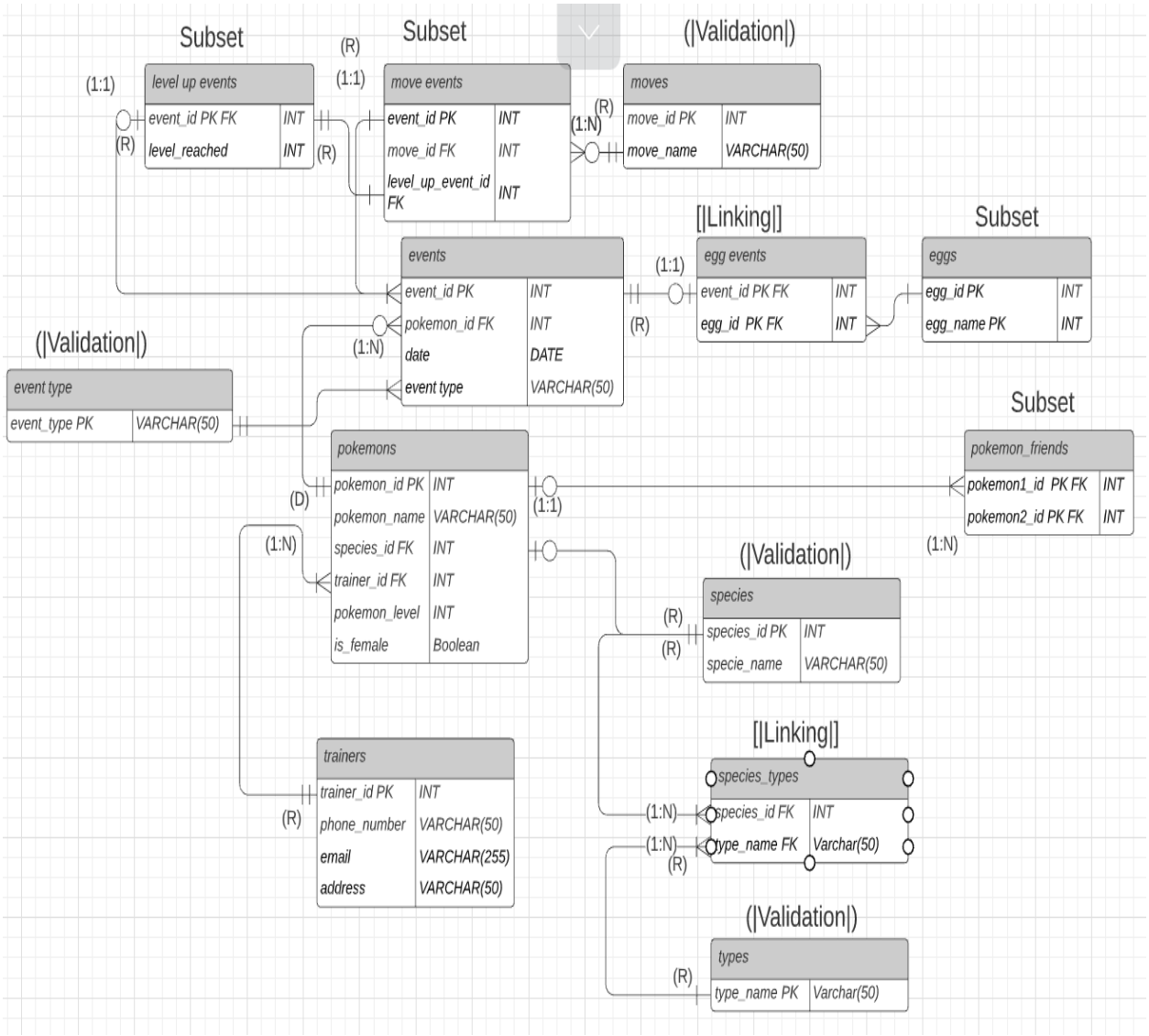
7. Pokemon can only have eggs if they are friend and owned by the same trainers

This will be implemented by checking if a friendship exists between the two Pokémon. It can be identified by checking the pokemon friendship table, once that relationship is established, we will check whether the trainer id associated with the two pokemon is the same. If both the statements are true only then we will allow the two pokemon to have an egg.

# APPENDIX

## Appendix 1

# ER diagram


Subset
Subset
(|Validation|)
level up events
event_id PK FK INT
level_reached INT
move events
event_id PK INT
move_id FK INT
level_up_event_id FK INT
moves
move_id PK INT
move_name VARCHAR(50)
(1:1)
(R)
(1:N)
[|Linking|]
Subset
events
event_id PK INT
pokemon_id FK INT
date DATE
event type VARCHAR(50)
egg events
event_id PK FK INT
egg_id PK FK INT
eggs
egg_id PK INT
egg_name PK INT
(|Validation|)
event type
event_type PK VARCHAR(50)
Subset
pokemons
pokemon_id PK INT
pokemon_name VARCHAR(50)
species_id FK INT
trainer_id FK INT
pokemon_level INT
is_female Boolean
pokemon_friends
pokemon1_id PK FK INT
pokemon2_id PK FK INT
(D)
(1:N)
(|Validation|)
species
species_id PK INT
specie_name VARCHAR(50)
trainers
trainer_id PK INT
phone_number VARCHAR(50)
email VARCHAR(255)
address VARCHAR(50)
[|Linking|]
species_types
species_id FK INT
type_name FK Varchar(50)
(|Validation|)
types
type_name PK Varchar(50)

**Appendix 2**

# *Field Specifications*

**Pokemon Trainers's Address**

"Keep track and locate the address of a pokemon's trainer."

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: address
Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: trainers
Label: address
Source Specification: _
Shared By: pokemons
Aliases: _
Description: An address stored in the trainers table that allows us to easily see where the address of each trainer is matched with their id and other information.

PHYSICAL ELEMENTS

Data Type: alphanumeric
Length: 1-50
Decimal Places: -
Input Mask: -
Display Format: -
Character Support:
Letters (A-Z): X
Numbers (0-9): X
Keyboard (.,/$#%): X
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: X
Primary:
Foreign: _
Alternate: _

Key Structure: Simple: _
Composite: _

Uniqueness: Non-unique: X
Unique: _

Null Support: Nulls OK: X
No nulls: _

Values Entered By: User: X
System: _

Required Value: No: X
Yes: _

Default Value: _

Range of Values: General

Edit Rule:
Enter now, edits allowed: X
Enter now, edits not allowed: _
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL X
+ _
- _
* _
/ _

NOTES:

**Is_Female**

**“**Identifies pokemon gender”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: is_female

Specification Type:
Unique: _
Generic: X
Replica: _
Parent Table: Pokemons
Label: is_female
Source Specification: _
Shared By: _
Aliases: _
Description: gender of the pokemon information

PHYSICAL ELEMENTS

Data Type: bool
Length: _
Decimal Places: _
Input Mask: _
Display Format: _
Character Support:
Letters (A-Z): _
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: X
Primary: _
Foreign: _
Alternate: _

Key Structure: Simple: _
Composite: _

Uniqueness: Non-unique: X
Unique: _

Null Support: Nulls OK: _
No nulls: X

Values Entered By: User: X
System: _

Required Value: No: _
Yes: X

Default Value: _

Range of Values: A-Z

Edit Rule:
Enter now, edits allowed: _
Enter now, edits not allowed: X
Enter later, edits allowed: _
Enter later, edits not allowed: _

Not determined at this time: _

Comparisons Allowed:
Same Field: X
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: X
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: X
ALL X
+ _
- _
* _
/ _

NOTES:

**Pokemon Level**

“Experience level of pokemon”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: pokemon_level

Specification Type:
Unique: _
Generic: _
Replica: X
Parent Table: pokemons
Label: pokemon_level
Source Specification: _
Shared By: _
Aliases: _
Description: experience level of the pokemon information

PHYSICAL ELEMENTS

Data Type: INT
Length: _
Decimal Places: _
Input Mask: _
Display Format: _
Character Support:
Letters (A-Z): _
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: X
Primary: _
Foreign: _
Alternate: _

Key Structure: Simple: _
Composite: _

Uniqueness: Non-unique: _
Unique: X

Null Support: Nulls OK: _
No nulls: X

Values Entered By: User: _
System: X

Required Value: No: _
Yes: X

Default Value: _

Range of Values: A-Z

Edit Rule:
Enter now, edits allowed: X
Enter now, edits not allowed: _
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: X
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: X
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: X
ALL X
+ _
- _
* _
/ _

NOTES:

**Pokemon Name**

“Name of the pokemon”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: pokemon_name

Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: Pokemons
Label: pokemon_name
Source Specification: _
Shared By: _
Aliases: _
Description: Name of the pokemon

PHYSICAL ELEMENTS

Data Type: Varchar(50)
Length: _
Decimal Places: _
Input Mask: _
Display Format: _
Character Support:
Letters (A-Z): X
Numbers (0-9): _
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: X
Primary: _
Foreign: _
Alternate: _

Key Structure: Simple: _
Composite: _

Uniqueness: Non-unique: X
Unique: _

Null Support: Nulls OK: _
No nulls: X

Values Entered By: User: X
System: _

Required Value: No: _
Yes: X

Default Value: _

Range of Values: A-Z

Edit Rule:
Enter now, edits allowed: _
Enter now, edits not allowed: X
Enter later, edits allowed: _
Enter later, edits not allowed: _

Not determined at this time: _

Comparisons Allowed:
Same Field: X
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: X
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL _
+ _
- _
* _
/ _

NOTES:

**Pokemon Species_ID**

“ID of pokemon species associated with Pokemon”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: species_ID

Specification Type:
Unique: _
Generic: _
Replica: X
Parent Table: Pokemon
Label: species_ID
Source Specification: _
Shared By: _
Aliases: _
Description: Id of the specie that is associated with its name

PHYSICAL ELEMENTS

Data Type: INT
Length: _
Decimal Places: _
Input Mask: _
Display Format: _
Character Support:
Letters (A-Z): _
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: _
Primary: _
Foreign: X
Alternate: _

Key Structure: Simple: X
Composite: _

Uniqueness: Non-unique: _
Unique: X

Null Support: Nulls OK: _
No nulls: X

Values Entered By: User: _
System: X

Required Value: No: _
Yes: X

Default Value: _

Range of Values: A-Z

Edit Rule:
Enter now, edits allowed: _
Enter now, edits not allowed: X
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: X
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: X
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL _
+ _
- _
* _
/ _

NOTES:

**Trainer_ID**

“Id of trainer associated with Pokemon”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: trainer_ID

Specification Type:
Unique: _
Generic: _
Replica: X
Parent Table: pokemons
Label: trainer_ID
Source Specification: _
Shared By: _
Aliases: _
Description: Id of the trainer that is associated with the pokemon information

PHYSICAL ELEMENTS

Data Type: INT
Length: _
Decimal Places: _
Input Mask: _
Display Format: _
Character Support:
Letters (A-Z): _
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: _
Primary: _
Foreign: X
Alternate: _

Key Structure: Simple: X
Composite: _

Uniqueness: Non-unique: _
Unique: X

Null Support: Nulls OK: _
No nulls: X

Values Entered By: User: _
System: X

Required Value: No: _
Yes: X

Default Value: _

Range of Values: A-Z

Edit Rule:
Enter now, edits allowed: _
Enter now, edits not allowed: X
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: X
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: X
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL _
+ _
- _
* _
/ _

NOTES:

**Event_ID**

“ID of event pokemon attended”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: event_id

Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: eggs
Label: _
Source Specification: _
Shared By: _
Aliases: _
Description: ID of event pokemon attend to practice move

PHYSICAL ELEMENTS

Data Type: INT
Length: _
Decimal Places: _
Input Mask: _
Display Format: _
Character Support:
Letters (A-Z): _
Numbers (0-9): _
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: _
Primary: X
Foreign: _
Alternate: _

Key Structure: Simple: _
Composite: _

Uniqueness: Non-unique: _
Unique: _

Null Support: Nulls OK: _
No nulls: _

Values Entered By: User: _
System: _

Required Value: No: _
Yes: _

Default Value: _

Range of Values: _

Edit Rule:
Enter now, edits allowed: _
Enter now, edits not allowed: _
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL _
+ _
- _
* _
/ _

NOTES:

**Father_ID**

“Id of father pokemon to new egg”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: father_id

Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: eggs
Label: _
Source Specification: _
Shared By: _
Aliases: _
Description: ID of father pokemon to new egg

PHYSICAL ELEMENTS

Data Type: INT
Length: _
Decimal Places: _
Input Mask: _
Display Format: _
Character Support:
Letters (A-Z): _
Numbers (0-9): _
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: _
Primary: _
Foreign: X
Alternate: _

Key Structure: Simple: _
Composite: _

Uniqueness: Non-unique: _
Unique: _

Null Support: Nulls OK: _
No nulls: _

Values Entered By: User: _
System: _

Required Value: No: _
Yes: _

Default Value: _

Range of Values: _

Edit Rule:
Enter now, edits allowed: _
Enter now, edits not allowed: _
Enter later, edits allowed: _
Enter later, edits not allowed: _

Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL _
+ _
- _
* _
/ _

NOTES:

**Mother_ID**

“ID of mother pokemon to new egg”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: mother_id

Specification Type:
Unique: _
Generic: _
Replica: _
Parent Table: eggs
Label: _
Source Specification: _
Shared By: _
Aliases: _
Description: ID of mother pokemon to new egg

PHYSICAL ELEMENTS

Data Type: INT
Length: _
Decimal Places: _
Input Mask: _
Display Format: _
Character Support:
Letters (A-Z): _
Numbers (0-9): _
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: _
Primary: _
Foreign: X
Alternate: _

Key Structure: Simple: _
Composite: _

Uniqueness: Non-unique: _
Unique: _

Null Support: Nulls OK: _
No nulls: _

Values Entered By: User: _
System: _

Required Value: No: _
Yes: _

Default Value: _

Range of Values: _

Edit Rule:
Enter now, edits allowed: _
Enter now, edits not allowed: _
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL _
+ _
- _
* _
/ _

NOTES:

**Event Date**

“Date pokemon attended event”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: event date

Specification Type:
Unique: _
Generic: _
Replica: _
Parent Table: events
Label: _
Source Specification: _
Shared By: _
Aliases: _
Description: Date pokemon attended event to learn moves

PHYSICAL ELEMENTS

Data Type: DATE
Length: _
Decimal Places: _
Input Mask: _
Display Format: _
Character Support:
Letters (A-Z): _
Numbers (0-9): _
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: X
Primary: _
Foreign: _
Alternate: _

Key Structure: Simple: _
Composite: _

Uniqueness: Non-unique: _
Unique: _

Null Support: Nulls OK: _
No nulls: _

Values Entered By: User: _
System: _

Required Value: No: _
Yes: _

Default Value: _

Range of Values: _

Edit Rule:
Enter now, edits allowed: _
Enter now, edits not allowed: _
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL _
+ _
- _
* _
/ _

NOTES:

**Level_Reached**

“Pokemon New Level after Leveling Up”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: level_reached

Specification Type:
Unique: _
Generic: _
Replica: _
Parent Table: level_ups
Label: _
Source Specification: _
Shared By: _
Aliases: _
Description: Level reached by pokemon after leveling up

PHYSICAL ELEMENTS

Data Type: INT
Length: _
Decimal Places: _
Input Mask: _
Display Format: _
Character Support:
Letters (A-Z): _
Numbers (0-9): _
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: _
Primary: _
Foreign: _
Alternate: _

Key Structure: Simple: _
Composite: _

Uniqueness: Non-unique: _
Unique: _

Null Support: Nulls OK: _
No nulls: _

Values Entered By: User: _
System: _

Required Value: No: _
Yes: _

Default Value: _

Range of Values: _

Edit Rule:
Enter now, edits allowed: _
Enter now, edits not allowed: _
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL _
+ _
- _
* _
/ _

NOTES:

**Move_ID**

“Number to represent move of a Pokemon”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: move_id

Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: moves
Label: move_id
Source Specification: _
Shared By:
Aliases:
Description: a unique id number to represent a move of a pokemon

PHYSICAL ELEMENTS

Data Type: numeric
Length:
Decimal Places: _
Input Mask: -
Display Format: -
Character Support:
Letters (A-Z): _
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: -
Primary: X
Foreign: _
Alternate: _

Key Structure: Simple: X
Composite: _

Uniqueness: Non-unique: _
Unique: X

Null Support: Nulls OK: X
No nulls: _

Values Entered By: User: _
System: X

Required Value: No: _
Yes: X

Default Value: _

Range of Values: General

Edit Rule:
Enter now, edits allowed: _
Enter now, edits not allowed: X
Enter later, edits allowed: _
Enter later, edits not allowed: _

Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL _
+ _
- _
* _
/ _

NOTES:

**Move_ID**

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: move_id

Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: moves
Label: move_id
Source Specification: _
Shared By:
Aliases:
Description: a unique id number to represent a move of a pokemon

PHYSICAL ELEMENTS

Data Type: numeric
Length:
Decimal Places: _
Input Mask: -
Display Format: -
Character Support:
Letters (A-Z): _
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: -
Primary: X
Foreign: _
Alternate: _

Key Structure: Simple: X
Composite: _

Uniqueness: Non-unique: _
Unique: X

Null Support: Nulls OK: X
No nulls: _

Values Entered By: User: _
System: X

Required Value: No: _
Yes: X

Default Value: _

Range of Values: General

Edit Rule:
Enter now, edits allowed: _
Enter now, edits not allowed: X
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL _
+ _
- _
* _
/ _

NOTES:

**Move_Name**
“Name of the move of the pokemon learned”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: move_name

Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: moves
Label: move_name
Source Specification: _
Shared By:
Aliases:
Description: the specific name of the move of the pokemon that each character has

PHYSICAL ELEMENTS

Data Type: alphanumeric
Length:
Decimal Places: 0
Input Mask: -
Display Format: -
Character Support:
Letters (A-Z): _
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: X
Primary: _
Foreign: _
Alternate: _

Key Structure: Simple: _
Composite: _

Uniqueness: Non-unique: X
Unique: _

Null Support: Nulls OK: X
No nulls: _

Values Entered By: User: _
System: X

Required Value: No: X
Yes: _

Default Value: N/A

Range of Values: General

Edit Rule:
Enter now, edits allowed: X
Enter now, edits not allowed: -
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL _
+ _
- _
* _
/ _

NOTES:

**Event_ID**

“Number associated with the event attended by the pokemon”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: event_id

Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: new moves
Label: event_id
Source Specification: _
Shared By: _
Aliases: _
Description: _

PHYSICAL ELEMENTS

Data Type: INT
Length: _
Decimal Places: _
Input Mask: _
Display Format: _
Character Support:
Letters (A-Z): _
Numbers (0-9): _
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: _
Primary: _
Foreign: _
Alternate: _

Key Structure: Simple: _
Composite: _

Uniqueness: Non-unique: _
Unique: _

Null Support: Nulls OK: _
No nulls: X

Values Entered By: User: _
System: X

Required Value: No: _
Yes: _

Default Value: _

Range of Values: _

Edit Rule:
Enter now, edits allowed: _
Enter now, edits not allowed: _
Enter later, edits allowed: _
Enter later, edits not allowed: _

Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL _
+ _
- _
* _
/ _

NOTES:

**Move_ID**

“Number associated with move of pokemon”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: move_id

Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: new moves
Label: _
Source Specification: _
Shared By: _
Aliases: _
Description: Number associated with the move of a pokemon

PHYSICAL ELEMENTS

Data Type: INT
Length: _
Decimal Places: _
Input Mask: _
Display Format: _
Character Support:
Letters (A-Z): _
Numbers (0-9): _
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: _
Primary: _
Foreign: _
Alternate: _

Key Structure: Simple: _
Composite: _

Uniqueness: Non-unique: _
Unique: _

Null Support: Nulls OK: _
No nulls: _

Values Entered By: User: _
System: _

Required Value: No: _
Yes: _

Default Value: _

Range of Values: _

Edit Rule:
Enter now, edits allowed: _
Enter now, edits not allowed: _
Enter later, edits allowed: _
Enter later, edits not allowed: _

Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL _
+ _
- _
* _
/ _

NOTES:

**Specie Name**

“Species name of a Pokemon”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: specie_name
Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: species
Label: specie_name
Source Specification: _
Shared By: pokemon_species, pokemons
Aliases: _
Description: A name representing the specific species of a pokemon.

PHYSICAL ELEMENTS

Data Type: alphanumeric
Length: 1-50
Decimal Places: -
Input Mask: -
Display Format: -
Character Support:
Letters (A-Z): x
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: X
Primary: -
Foreign: _
Alternate: _

Key Structure: Simple: -
Composite: _

Uniqueness: Non-unique: X
Unique: -

Null Support: Nulls OK: X
No nulls: _

Values Entered By: User: X
System: _

Required Value: No: X
Yes: _

Default Value: _

Range of Values: General

Edit Rule:
Enter now, edits allowed: X
Enter now, edits not allowed: _
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL X
+ _
- _
* _
/ _

NOTES:

**Species ID**

“Number identifying a pokemon species"

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: species_id
Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: pokemon_species
Label: species_id PK
Source Specification: _
Shared By: species_types, pokemons
Aliases: _
Description: A unique INT representing a pokemon species

PHYSICAL ELEMENTS

Data Type: exact number
Length:
Decimal Places: 0
Input Mask: -
Display Format: -
Character Support:
Letters (A-Z): _
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: _
Primary: X
Foreign: -
Alternate: _

Key Structure: Simple: X
Composite: _

Uniqueness: Non-unique: -
Unique: X

Null Support: Nulls OK: _
No nulls: X

Values Entered By: User: X
System: _

Required Value: No: _
Yes: X

Default Value: _

Range of Values: -

Edit Rule:
Enter now, edits allowed: -
Enter now, edits not allowed: X
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL X
+ _
- _
* _
/ _

NOTES:

**Type Name**

"Number representing a Type of Pokemon"

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: type_name
Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: pokemon_type
Label: type_name PK
Source Specification: _
Shared By: pokemon_types
Aliases: _
Description: The type of pokemon representing by a INT. This allows us to see what type of Pokemon we have as well what other kinds of pokemon he relates to

PHYSICAL ELEMENTS

Data Type: exact number
Length: -
Decimal Places: -
Input Mask: -
Display Format: -
Character Support:
Letters (A-Z): -
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: -
Primary: X
Foreign: _
Alternate: _

Key Structure: Simple: X
Composite: _

Uniqueness: Non-unique: _
Unique: X

Null Support: Nulls OK: _
No nulls: X

Values Entered By: User: X
System: _

Required Value: No: _
Yes: X

Default Value: _

Range of Values: General

Edit Rule:
Enter now, edits allowed: _
Enter now, edits not allowed: X
Enter later, edits allowed: _
Enter later, edits not allowed: _

Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL X
+ _
- _
* _
/ _

NOTES:

**Trainer Email**

“Email of Pokemon trainer”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: email
Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: trainers
Label: email
Source Specification: _
Shared By: pokemon
Aliases: _
Description: The email of each trainer allowing us to easily have contact information stored and matched with a trainer.

PHYSICAL ELEMENTS

Data Type: alphanumeric
Length: 1-255
Decimal Places: -
Input Mask: -
Display Format: -
Character Support:
Letters (A-Z): X
Numbers (0-9): X
Keyboard (.,/$#%): X
Special (©®™Σπ): X

LOGICAL ELEMENTS

Key Type: Non: X
Primary: _
Foreign: _
Alternate: _

Key Structure: Simple: _
Composite: _

Uniqueness: Non-unique: _
Unique: _

Null Support: Nulls OK: X
No nulls: _

Values Entered By: User: X
System: _

Required Value: No: X
Yes: _

Default Value: _

Range of Values: General

Edit Rule:
Enter now, edits allowed: X
Enter now, edits not allowed: _
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL X
+ _
- _
* _
/ _

NOTES:

**Trainer Phone_Number**

“Phone Number of Pokemon Trainer”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: phone_number
Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: trainers
Label: phone_number
Source Specification: _
Shared By: pokemons
Aliases: _
Description: The phone number of each trainer allowing easy access to retireve a number from a trainer.

PHYSICAL ELEMENTS

Data Type: numeric
Length:
Decimal Places: -
Input Mask: -
Display Format: -
Character Support:
Letters (A-Z): -
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: X
Primary: _
Foreign: _
Alternate: _

Key Structure: Simple: _
Composite: _

Uniqueness: Non-unique: X
Unique: _

Null Support: Nulls OK: X
No nulls: _

Values Entered By: User: X
System: _

Required Value: No: X
Yes: _

Default Value: _

Range of Values: General

Edit Rule:
Enter now, edits allowed: X
Enter now, edits not allowed: _
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL X
+ _
- _
* _
/ _

NOTES:

**Trainer_ID**

“Number associated with Pokemon Trainer”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: trainer_id
Specification Types:
Unique: X
Generic: _
Replica: _
Parent Table: trainers
Label: trainer_id PK
Source Specification: _
Shared By: species_types
Aliases: _
Description: This is a unique number given to a trainer to identify them easily allowing us to keep track of data for the trainer.

PHYSICAL ELEMENTS

Data Type: exact number
Length: -
Decimal Places: -
Input Mask: -
Display Format: -
Character Support:
Letters (A-Z): -
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: -
Primary: X
Foreign: _
Alternate: _

Key Structure: Simple: X
Composite: _

Uniqueness: Non-unique: _
Unique: X

Null Support: Nulls OK: -
No nulls: X

Values Entered By: User: -
System: X

Required Value: No: _
Yes: X

Default Value: _

Range of Values: General

Edit Rule:
Enter now, edits allowed: -
Enter now, edits not allowed: X
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL X
+ _
- _
* _
/ _

NOTES:

**Type Name**

“Type of species pokemon is”

FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: type_name PK
Specification Type:
Unique: X
Generic: _
Replica: _
Parent Table: types
Label: type_name PK
Source Specification: _
Shared By: species_types
Aliases: _
Description: This represents the type of species the pokemon is. This allows us to group the pokemon by their type of species.

PHYSICAL ELEMENTS

Data Type: exact number
Length: -
Decimal Places: -
Input Mask: -
Display Format: -
Character Support:
Letters (A-Z): -
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: -
Primary: X
Foreign: _
Alternate: _

Key Structure: Simple: X
Composite: _

Uniqueness: Non-unique: _
Unique: X

Null Support: Nulls OK: -
No nulls: X

Values Entered By: User: -
System: X

Required Value: No: _
Yes: X

Default Value: _

Range of Values: General

Edit Rule:

Enter now, edits allowed: -
Enter now, edits not allowed: X
Enter later, edits allowed: _
Enter later, edits not allowed: _

Not determined at this time: _

Comparisons Allowed:
Same Field: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: _
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: _
ALL X
+ _
- _
* _
/ _

NOTES:

**Pokemon1_id**
**“ID of one of pokemon’s friends”**
FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: pokemon1_id

Specification Type:
Unique: _
Generic: X
Replica: _
Parent Table: Pokemons
Label: pokemon1_id
Source Specification: _
Shared By: _
Aliases: _
Description: id of the pokemon’s friend information

PHYSICAL ELEMENTS

Data Type: int
Length:
Decimal Places: 0
Input Mask: _
Display Format: _
Character Support:
Letters (A-Z): _
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: -
Primary: X
Foreign: X
Alternate: _

Key Structure: Simple: X
Composite: _

Uniqueness: Non-unique: -
Unique: X

Null Support: Nulls OK: _
No nulls: X

Values Entered By: User: X
System: _

Required Value: No: X
Yes: -

Default Value: _

Range of Values: A-Z

Edit Rule:
Enter now, edits allowed: X
Enter now, edits not allowed: -
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: X
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: X
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: X
ALL X
+ _
- _
* _
/ _

NOTES:

**Pokemon2_ID**
**"ID of one of pokemon's friends"**
FIELD SPECIFICATIONS

GENERAL ELEMENTS

Field Name: pokemon2_id

Specification Type:
Unique: _
Generic: X
Replica: _
Parent Table: Pokemon_friends, pokemon
Label: pokemon2_id
Source Specification: _
Shared By: _
Aliases: _
Description: id of the pokemon's friend information

PHYSICAL ELEMENTS

Data Type: int
Length:
Decimal Places: 0
Input Mask: _
Display Format: _
Character Support:
Letters (A-Z): _
Numbers (0-9): X
Keyboard (.,/$#%): _
Special (©®™Σπ): _

LOGICAL ELEMENTS

Key Type: Non: -
Primary: X
Foreign: X
Alternate: _

Key Structure: Simple: X
Composite: _

Uniqueness: Non-unique: -
Unique: X

Null Support: Nulls OK: _
No nulls: X

Values Entered By: User: X
System: _

Required Value: No: X
Yes: -

Default Value: _

Range of Values: A-Z

Edit Rule:
Enter now, edits allowed: X
Enter now, edits not allowed: -
Enter later, edits allowed: _
Enter later, edits not allowed: _
Not determined at this time: _

Comparisons Allowed:
Same Field: X
ALL X
= _
> _
>= _
!= _
< _
<= _

Other Fields: _
ALL _
= _
> _
>= _
!= _
< _
<= _

Value Expr.: X
ALL X
= _
> _
>= _
!= _
< _
<= _

Operations Allowed:
Same Field: _
ALL _
+ _
- _
* _
/ _

Other Fields: _
ALL _
+ _
- _
* _
/ _

Value Expr.: X
ALL X
+ _
- _
* _
/ _

NOTES:

*Appendix 3*

# *Business Rules*

**“Trainer can only have 2 pokemons at once”**

BUSINESS RULE SPECIFICATIONS

RULE INFORMATION

Statement: Trainer can only have 2 pokemons at once

Constraint: Count the number of pokemons related to a specific trainer_id

Type: Database Oriented: X
Application Oriented _

Category: Field Specific: X
Relationship Specific: _

Test On: Insert: X
Delete: _
Update: _

STRUCTURES AFFECTED

Field Names: _

Table Names: _

FIELD ELEMENTS AFFECTED

Physical Elements: Data Type: _
Length: _
Decimal Places: _
Character Support: _
Input Mask: _
Display Format: _

Logical Elements: Key Type: _
Key Structure: _
Uniqueness: _
Null Support: _
Values Entered By: user
Default Value: _
Range of Values: 0-2
Comparisons Allowed: X
Operations Allowed: _
Edit Rule: enter later, edits allowed

ACTION TAKEN

We check the number of pokemons ids related to a trainer id and do not let more entries if count exceeds 2.
_

NOTES: _

**“Pokemons cannot learn a move that they have already learnt”**

BUSINESS RULE SPECIFICATIONS

RULE INFORMATION

Statement: Pokemons cannot learn a move that they have already learnt

Constraint: Check the moves that pokemons have learnt using the events table and their pokemon id, if move already exist then no new event added

Type: Database Oriented: X
Application Oriented _

Category: Field Specific: X
Relationship Specific: _

Test On: Insert: X
Delete: _
Update: _

STRUCTURES AFFECTED

Field Names: _

Table Names: _

FIELD ELEMENTS AFFECTED

Physical Elements: Data Type: _
Length: _
Decimal Places: _
Character Support: _
Input Mask: _
Display Format: _

Logical Elements: Key Type: _
Key Structure: _
Uniqueness: _
Null Support: _
Values Entered By: user
Default Value: _
Range of Values: _
Comparisons Allowed: X
Operations Allowed: _
Edit Rule: enter now, edits not allowed

ACTION TAKEN

If pokemon has a move that they already learnt it cannot be added as a new event
_

NOTES: _

**“2 or more people cannot have the same phone number”**

BUSINESS RULE SPECIFICATIONS

RULE INFORMATION

Statement: No same phone numbers

Constraint: Two trainers cannot have the same phone numbers, otherwise we will be unable to reach them on their phone. We check if phone number already
exists in the current table of trainers, if not then number is eligible

Type: Database Oriented: X
Application Oriented _

Category: Field Specific: X
Relationship Specific: _

Test On: Insert: X
Delete: _
Update: _

STRUCTURES AFFECTED

Field Names: _

Table Names: _

FIELD ELEMENTS AFFECTED

Physical Elements: Data Type: _
Length: _
Decimal Places: _
Character Support: _
Input Mask: _
Display Format: _

Logical Elements: Key Type: _
Key Structure: _
Uniqueness: X
Null Support: _
Values Entered By: user
Default Value: _
Range of Values: _
Comparisons Allowed: X
Operations Allowed: _
Edit Rule: enter now, edit later

ACTION TAKEN

_ User is not allowed to enter the same phone number, if another trainer has the same phone number.

NOTES: _

**“Egg should have male and female parents”**

BUSINESS RULE SPECIFICATIONS

RULE INFORMATION

Statement: Egg should have male and female parents.

Constraint: Male pokemon and female pokemon has to go to male and female pokeon field of the egg table

Type: Database Oriented: X
Application Oriented _

Category: Field Specific: _
Relationship Specific: X

Test On: Insert: X
Delete: _
Update: _

STRUCTURES AFFECTED

Field Names: father_id, mother_id

Table Names: egg table

FIELD ELEMENTS AFFECTED

Physical Elements: Data Type: _
Length: _
Decimal Places: _
Character Support: _
Input Mask: _
Display Format: _

Logical Elements: Key Type: _
Key Structure: _
Uniqueness: _
Null Support: _
Values Entered By: _
Default Value: _
Range of Values: _
Comparisons Allowed: X
Operations Allowed: _
Edit Rule: Edit not arrowed

ACTION TAKEN

_ Repeat

NOTES: We will look at mother_id and father_id and then check the pokemon details by accessing their information in the pokemon tables. Mother_id has to be a female meaning that the is_female boolean has to be true and the pokemon related to father_id must have a is_female as false

**“A Date for an event must come after all previous dates or equal the last one”**

BUSINESS RULE SPECIFICATIONS

RULE INFORMATION

Statement: A Date entered for an event cannot come before the previous date entered for an event.

Constraint: The date must be greater or equal to the date.

Type: Database Oriented: X
Application Oriented _

Category: Field Specific: X
Relationship Specific: -

Test On: Insert: X
Delete: _
Update: _

STRUCTURES AFFECTED

Field Names: event_id pk

Table Names: required values, comparisons allowed

FIELD ELEMENTS AFFECTED

Physical Elements: Data Type: _
Length: _
Decimal Places: _
Character Support: _
Input Mask: _
Display Format: _

Logical Elements: Key Type: _
Key Structure: _
Uniqueness: _
Null Support: _
Values Entered By: _
Default Value: _
Range of Values: _
Comparisons Allowed: X
Operations Allowed: _
Edit Rule: Enter now, edits not allowed

ACTION TAKEN

Operations allowed was set to 'YES', and Edit Rule was set to "Enter now, edits not allowd"

NOTES: _

**“No Pokémon can have an egg if they are not friends and don’t both have the same trainer.”**

BUSINESS RULE SPECIFICATIONS

RULE INFORMATION

Statement: **“**No Pokémon can have an egg if they are not friends and don’t both have the same trainer.**”**

Constraint: This will be implemented by checking if a friendship exists between the two Pokémon. It can be identified by checking the pokemon friendship table, once that relationship is established, we will check whether the trainer id associated with the two pokemon is the same. If both the statements are true only then we will allow the two pokemon to have an egg

Type: Database Oriented: X
Application Oriented _

Category: Field Specific: _
Relationship Specific: X

Test On: Insert: X
Delete: _
Update: _

STRUCTURES AFFECTED

Field Names:egg_id

Table Names: eggs

FIELD ELEMENTS AFFECTED

Physical Elements: Data Type: _
Length: _
Decimal Places: _
Character Support: _
Input Mask: _
Display Format: _

Logical Elements: Key Type: _
Key Structure: _
Uniqueness: _
Null Support: _
Values Entered By: _
Default Value: _
Range of Values: _
Comparisons Allowed: _
Operations Allowed: _
Edit Rule: Edit not arrowed

ACTION TAKEN

_ Check Pokemon friends table and see if they are friends

NOTES: _

**“Level Cannot Go Down”**

BUSINESS RULE SPECIFICATIONS

RULE INFORMATION

Statement: Level cannot go down.

Constraint: New level insert should not record a level lower or equal to the last updated level.

Type: Database Oriented: X
Application Oriented _

Category: Field Specific: X
Relationship Specific: _

Test On: Insert: X
Delete: _
Update: _

STRUCTURES AFFECTED

Field Names: level_reached

Table Names: level ups

FIELD ELEMENTS AFFECTED

Physical Elements: Data Type: _
Length: _
Decimal Places: _
Character Support: _
Input Mask: _
Display Format: _

Logical Elements: Key Type: _
Key Structure: _
Uniqueness: _
Null Support: _
Values Entered By: _
Default Value: _
Range of Values: _
Comparisons Allowed: _
Operations Allowed: _
Edit Rule: Edit not arrowed

ACTION TAKEN

_ Repeat

NOTES: _