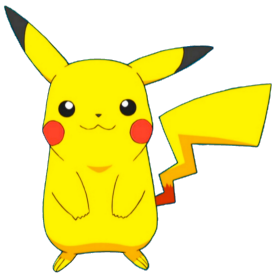
University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: Oct. 25, 2021

Group Number: 84

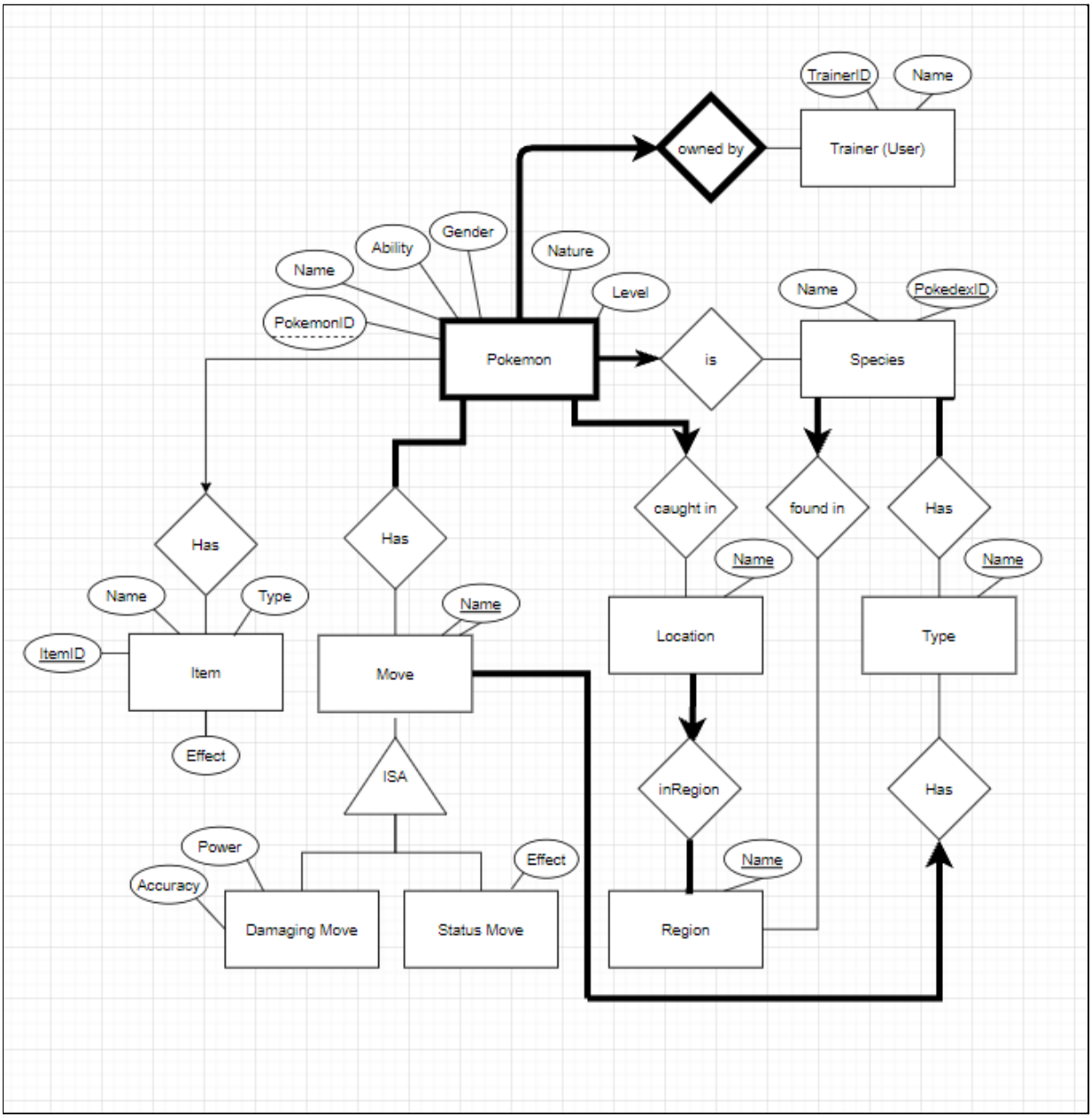


Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Sarah Gu	29720588	q3f3b	sarahyzgu@gmail.com
Philly Tan	79954962	j6v2b	philly@phillytan.xyz
Janelle Wong	65602740	d1g3b	janellewong112@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

ER DIAGRAM



Changes Made:

- Added ability attribute to Pokemon for additional FDs
- Weak relationship between Pokemon and Species was removed because we realized PokemonID did not need PokedexID to be a unique key and did not depend on PokedexID.
- Added Type and ItemID (new key) to Item
- Moved HasType relationship from Pokemon to Species because we realized that the type of the Pokemon should be associated with its species instead of directly to a Pokemon

Schema / Functional Dependencies / SQL DDL / Table

Underline = Primary Key
Bold = Foreign Key
** We have no candidate keys

Trainer(TrainerID, Name)
TrainerID → Name

```
CREATE TABLE Trainer (
      TrainerID INTEGER PRIMARY KEY,
      Name CHAR(50) NOT NULL
)
```

```
INSERT INTO Trainer (TrainerID, Name)
VALUES (1, "Joe Smith"),
      (2, "Tom Smithson"),
      (3, "Johnny Small"),
      (4, "Jerry Gerry"),
      (5, "Sally Johnson");
```

Item(Item ID, Name, Type, Effect)

ItemID → Name, Type, Effect

Name, Type → Effect (This violates BCNF and 3NF because (Name, Type) does not contain superkey and (Effect) does not contain part of a key)

Decompose into:

Item1(ItemID, Name, Type)

ItemID → Name, Type

Item2(Name, Type, Effect)

Name, Type → Effect

```
CREATE TABLE Item1 (
      ItemID CHAR(50) PRIMARY KEY,
      Name CHAR(50),
      Type CHAR(50)
)
```

```
INSERT INTO Item1 (ItemID, Name, Type)
VALUES (1, "Oran Berry", "Berry"),
       (2, "Cheri Berry", "Berry"),
       (3, "Master Ball", "Pokeball"),
       (4, "Great Ball", "Pokeball"),
       (5, "Splash Plate", "Plate");
```

```
CREATE TABLE Item2 (
      Name CHAR(50),
      Type CHAR(50),
      Effect CHAR(50),
      PRIMARY KEY (Name, Type)
)
```

```
INSERT INTO Item2 (Name, Type, Effect)
VALUES ("Oran Berry", "Berry", "Restores 10 HP"),
       ("Cheri Berry", "Berry", "Cures Paralysis"),
       ("Master Ball", "Pokeball", "Catches Pokemon with 100% success"),
       ("Great Ball", "Pokeball", "Catches Pokemon with improved success"),
       ("Splash Plate", "Plate", "Increases Water-type power");
```

Pokemon(PokemonID, Name, Gender, Nature, Level, Ability, **ItemID**, **TrainerID**, **PokedexID**, **LocationName**)

TrainerID, PokemonID → Name, Gender, Nature, Level, Ability, PokedexID, LocationName, ItemID

PokedexID → Ability (This violates BCNF and 3NF because (PokedexID) does not contain superkey and (Ability) does not contain part of a key)

Decompose into:

Pokemon1(PokemonID, Name, Gender, Nature, Level, **ItemID**, **TrainerID**, **PokedexID**, **LocationName**)

TrainerID, PokemonID → Name, Gender, Nature, Level, Ability, PokedexID, LocationName, ItemID

Pokemon2(PokedexID, Ability)

PokedexID → Ability

```
CREATE TABLE Pokemon1 (
      PokemonID INTEGER,
      Name CHAR(50),
      Gender CHAR(50),
      Nature CHAR(50),
      Level INTEGER,
      ItemID INTEGER,
      TrainerID INTEGER,
      PokedexID INTEGER,
      LocationName CHAR(50),
      PRIMARY KEY (PokemonID, TrainerID),
      FOREIGN KEY (ItemID) REFERENCES Item,
      FOREIGN KEY (TrainerID) REFERENCES Trainer,
      FOREIGN KEY (PokedexID) REFERENCES Species,
      FOREIGN KEY (LocationName) REFERENCES Location
)
```

```
INSERT INTO  Pokemon1 (Pokemon ID, Name, Gender, Nature, Level, ItemID,
TrainerID, PokedexID, LocationName)
VALUES      (1, "Sparky", "Female", "Brave", 40, 1, 2, 25, "Pallet Town"),
(2, "Flamey", "Male", "Adamant", 20, 2, 1, 4, "Viridian City"),
(3, "Leafy", "Female", "Calm", 6, NULL, 5, 1, "Route 3"),
(4, "Splashy", "Female", "Lonely", 10, 5, 2, 7, "Viridian City"),
(5, "Tree", "Male", "Naive", 80, NULL, 4, 3, "Pallet Town");


CREATE TABLE Pokemon2(
      PokedexID INTEGER PRIMARY KEY,
      Ability CHAR(50)
)


INSERT INTO  Pokemon2 (PokedexID, Ability)
VALUES (25, "Static"),
       (1, "Overgrow"),
       (3, "Overgrow"),
       (4, "Blaze"),
       (7, "Torrent");
```

Species(PokedexID, Name, **RegionName**)

PokedexID → Name, RegionName

```
CREATE TABLE Species (
      PokedexID INTEGER PRIMARY KEY,
      Name CHAR(50) NOT NULL,
      FOREIGN KEY (RegionName) REFERENCES Region
)


INSERT INTO Species (PokedexID, Name, Region)
VALUES (1, "Bulbasaur", "Kanto"),
       (3, "Venusaur", "Kanto"),
       (4, "Charmander", "Kanto"),
       (7, "Squirtle", "Kanto"),
       (25, "Pikachu", "Kanto");
```

Type(Name)

Name → Name

```
CREATE TABLE Type (
      Name CHAR(50) PRIMARY KEY
)
```

```
INSERT INTO Type (Name)
VALUES ("Normal"), ("Grass"), ("Fire"), ("Water"), ("Electric");
```

Location(Name, **RegionName**)

Name → RegionName

```
CREATE TABLE Location (
      Name CHAR(50) PRIMARY KEY,
      RegionName CHAR(50),
      FOREIGN KEY (RegionName) REFERENCES Region
)
```

```
INSERT INTO Location (Name, RegionName)
VALUES ("Pallet Town", "Kanto"), ("Viridian City", "Kanto"), ("Route 3",
"Kanto"), ("Spear Pillar", "Sinnoh"), ("Jubilife City", "Sinnoh");
```

Region(RegionName)

RegionName → RegionName

```
CREATE TABLE Region (
      RegionName CHAR(50) PRIMARY KEY
)
```

```
INSERT INTO Region (RegionName)
VALUES ("Kanto"), ("Johto"), ("Hoenn"), ("Sinnoh"), ("Unova");
```

Move(Name, Type, Power, Accuracy, Effect)
Name → Type, Power, Accuracy, Effect

NOTE: Creating one table instead of three separate tables as assuming we'll be using a Total + Overlap ISA.

```
CREATE TABLE Move (
      Name CHAR(50) PRIMARY KEY,
      Type CHAR(50),
      Power CHAR(50),
      Accuracy CHAR(50),
      Effect CHAR(50),
      FOREIGN KEY (Type) REFERENCES Type
)
```

```
INSERT INTO Move (Name, Type, Power, Accuracy, Effect)
VALUES ("Tackle", "Normal", 40, 100, NULL), ("Water Gun", "Water", 40, 100,
NULL), ("Thunderbolt", "Electric", 90, 100, NULL), ("Fire Blast", "Fire",
110, 85, NULL), ("Razor Leaf", "Grass", 55, 95, NULL);
```

SpeciesHasType(**PokedexID**, **TypeName**)
PokedexID → TypeName

```
CREATE TABLE SpeciesHasType (
      PokedexID INTEGER,
      TypeName CHAR(50),
      PRIMARY KEY (PokedexID, TypeName),
      FOREIGN KEY (PokedexID) REFERENCES Species,
      FOREIGN KEY (TypeName) REFERENCES Type
)
```

```
INSERT INTO SpeciesHasType (PokedexID, TypeName)
VALUES (1, "Grass"), (3, "Grass"), (4, "Fire"), (7, "Water"), (25,
"Electric");
```

PokemonHasMove(<u>PokemonID</u>, <u>TrainerID</u>, <u>MoveName</u>)
PokemonID, TrainerID → MoveName

```
CREATE TABLE PokemonHasMove(
      PokemonID INTEGER,
      TrainerID CHAR(50),
      PRIMARY KEY (PokemonID, TrainerID, MoveName),
      FOREIGN KEY (PokemonID) REFERENCES Pokemon,
      FOREIGN KEY (TrainerID) REFERENCES Trainer,
      FOREIGN KEY (MoveName) REFERENCES Move
)
```

```
INSERT INTO PokemonHasMove (PokemonID, TrainerID, MoveName)
VALUES (1, 2, "Thunderbolt"),
       (2, 1 "Fire Blast"),
       (3, 5, "Razor Leaf"),
       (4, 2, "Water Gun"),
       (5, 4, "Razor Leaf");
```

Sample Queries

Filter: Filter out all my pokemon by the Water type.

Insertion: Add a pokemon to in a trainer's list by giving the pokemon the TrainerID

Deletion: Remove a pokemon in a trainer's list

Sort: Sort all the pokemon from one trainer by highest to lowest levels

Selection: Find all the pokemon from a certain region
Selection: Find all pokemon with an accuracy damaging move

Update: Change the name or details (moveset) of a pokemon

Join: Find all Fire type pokemon that have a damaging power move

Projection: Finding specific attributes associated with a particular pokemon

Aggregation with Group By: Find the average level of all Lighting type pokemon in the trainer's list

Nested Aggregation with Group By: Find the max level pokemon in each type in the trainer's list

Division: Divide the number of Leaf type pokemon with all pokemon in the Trainer's list to see the percentage of Leaf type pokemon in the list