University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: __2_____

Date: __21/07/2025_____

Group Number: ____36_____

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Graydon Strachan	37275377	gstracha	glstrachan@outlook.com
Anthony Lu	35658681	alu34	anthonylu7678@gmail.com
Dikpaal Patel	37647864	dikpaal	dikpaalpatel123@gmail.com

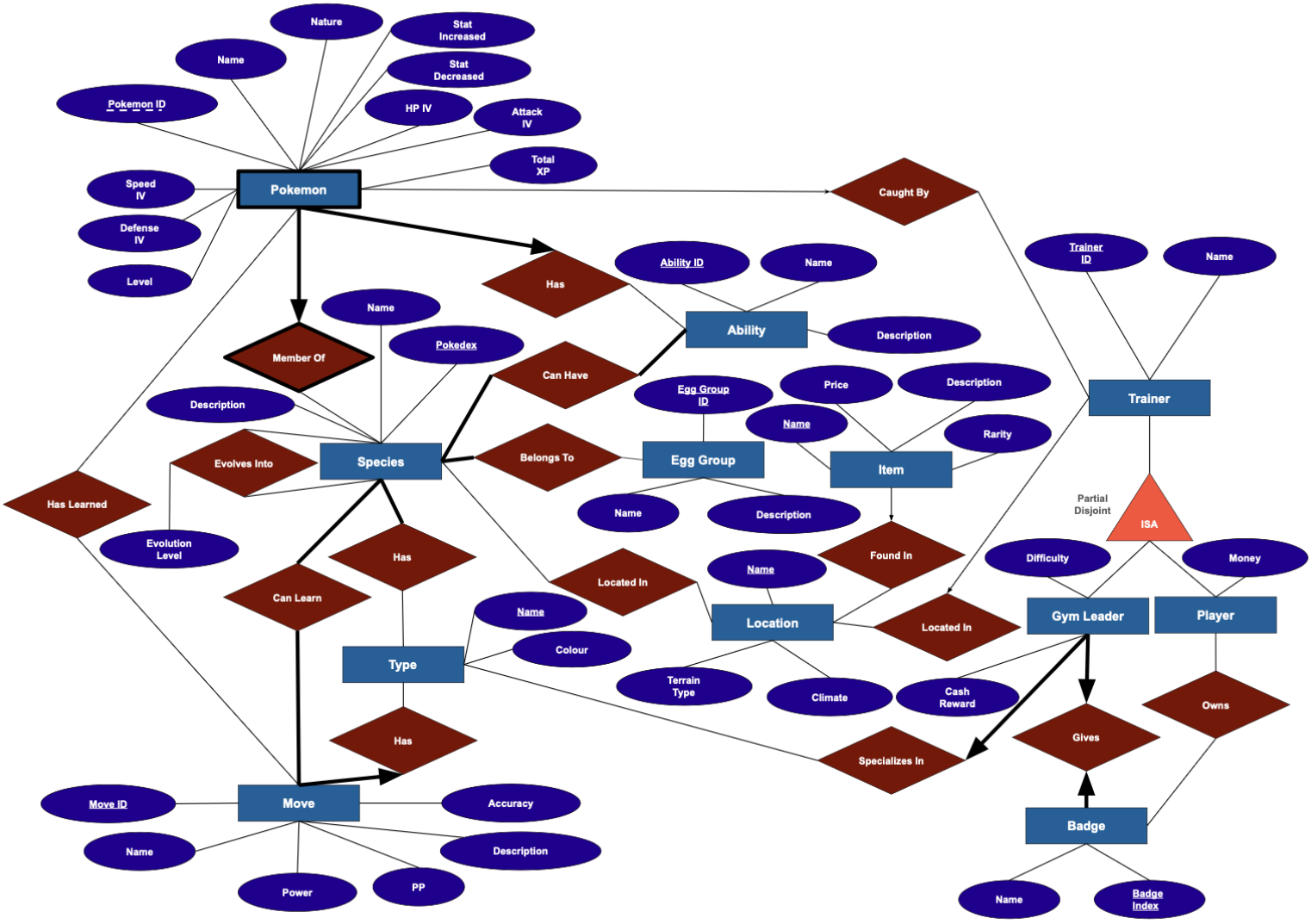
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address,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

2) Brief Summary

Our application is meant to be a Pokemon game management lookup system. The functionality that the database will provide is the ability to quickly check the relationships between different Pokemon, Items, Locations and Trainers. More specifically, the database will provide the user with the ability to check which Pokemon species evolves into which other species, which species and items can be found at each location, which moves can be learned by which Pokemon species, and which Pokemon each trainer has.

3) ER Diagram



Notes:

- We changed the wording of Pokemon “Base” stats to “IV”s (individual values) to better reflect that we are modelling that specific aspect of each Pokemon instance.
- Jessie suggested that we could potentially use existing attributes as primary keys instead of generic IDs. However, we decided to keep most IDs as primary keys, just because these IDs are already built into the world of Pokemon, so we felt that keeping these IDs would more accurately capture the characteristics of entities.
- We added attributes (such as nature, stat_increased, and stat_decreased to Pokemon) in order to create more non-trivial, non-key-related functional dependencies.
- We added a relationship between Gym Leader and Type.
- We added a badge entity and removed certain badge-related attributes to improve the structure of the database.

4) The schema derived from your ER diagram

Notes:

- Primary keys are underlined
- Foreign keys are bolded
- Unless specified, all table fields cannot be null
- Every primary key is also a candidate key
- From here on out, many names have been changed for tables modelling many-to-many relationships, since most of the names given to these relationships in the ER diagram are quite similar. For example, there are many “Has” relationships in our diagram. In order to clearly indicate which relationship we are talking about, we have mostly incorporated the names of two entities participating in the relationship into the relationships’ table names. For example, the table for the “Has” relationship between Species and Type has been named “Species_Has_Type”.

Entities and One-to-Many Relationships:

Pokemon(**pokedex**: INTEGER, pokemon_id: INTEGER, name: VARCHAR(20), level: INTEGER, total_XP: INTEGER, nature: VARCHAR(12), stat_increased: VARCHAR(12), stat_decreased: VARCHAR(12), HP_IV: INTEGER, attack_IV: INTEGER, defense_IV: INTEGER, speed_IV: INTEGER, **ability_id**: INTEGER, **trainer_id**: INTEGER)

- trainer_id can be null
- FOREIGN KEY (pokedex) REFERENCES Species(pokedex)
- FOREIGN KEY (ability_id) REFERENCES Ability(ability_id)

- FOREIGN KEY (trainer_id) REFERENCES Trainer(trainer_id)

Ability(ability_id: INTEGER, name: VARCHAR(255), description: VARCHAR(1000))
- name is a candidate key

Species(pokedex: INTEGER, name: VARCHAR(12), description:VARCHAR(1000))
- name is a candidate key

Egg_Group(egg_group_id: INTEGER, name: VARCHAR(30), description: VARCHAR(1000));
- name is a candidate key

Item(name: VARCHAR(255), description: VARCHAR(1000), price: INTEGER, rarity: VARCHAR(10), **location_name**: VARCHAR(40))
- price, rarity, and location_name can be null
- FOREIGN KEY location_name REFERENCES Location(name)

Type(name: VARCHAR(9), colour: VARCHAR(50))
- colour is a candidate key

Location(name: VARCHAR(40), climate: VARCHAR(255), terrain_type: VARCHAR(255))

Trainer(trainer_id: INTEGER, name: VARCHAR(255), **location_name**: VARCHAR(40))
- location_name can be null
- FOREIGN KEY location_name REFERENCES Location(name)

Gym_Leader(**trainer_id**: INTEGER, difficulty: VARCHAR(10), cash_reward: INTEGER, **specialty_type_name**: VARCHAR(9), **badge_index**: INTEGER)
- badge_index is unique
- badge_index is a candidate key
- FOREIGN KEY (trainer_id) REFERENCES Trainer(trainer_id)
- FOREIGN KEY (specialty_type_name) REFERENCES Type(name)
- FOREIGN KEY (badge_index) REFERENCES Badge(badge_index)

Player(**trainer_id**: INTEGER, money: INTEGER)
- FOREIGN KEY (trainer_id) REFERENCES Trainer(trainer_id)

Move(move_id: INTEGER, name: VARCHAR(50), power: INTEGER, pp: INTEGER, accuracy: INTEGER, description: VARCHAR(1000), **type_name**: VARCHAR(9))

- name is a candidate key
- FOREIGN KEY (type_name) REFERENCES Type(name)

Badge(badge_index: INTEGER, name: VARCHAR(15))
- name is a candidate key

Many-to-Many Relationships:

Species_Evolves_Into(**old_pokedex**: INTEGER, **new_pokedex**: INTEGER, evolution_level: INTEGER);
- evolution_level can be null
- FOREIGN KEY (old_pokedex) REFERENCES Species(pokedex)
- FOREIGN KEY (new_pokedex) REFERENCES Species(pokedex)

Pokemon_Has_Learned_Move(**pokedex**: INTEGER, **pokemon_id**: INTEGER, **move_id**: INTEGER)
- FOREIGN KEY (pokedex, pokemon_id) REFERENCES Pokemon(pokedex, pokemon_id)
- FOREIGN KEY (move_id) REFERENCES Move(move_id)

Species_Has_Type(**pokedex**: INTEGER, **type_name**: VARCHAR(9));
- FOREIGN KEY (pokedex) REFERENCES Species(pokedex)
- FOREIGN KEY (type_name) REFERENCES Type(name)

Species_Can_Learn_Move(**pokedex**: INTEGER, **move_id**: INTEGER)
- FOREIGN KEY (pokedex) REFERENCES Species(pokedex)
- FOREIGN KEY (move_id) REFERENCES Move(move_id)

Species_Can_Have_Ability(**pokedex**: INTEGER, **ability_id**: INTEGER);
- FOREIGN KEY (pokedex) REFERENCES Species(pokedex)
- FOREIGN KEY (ability_id) REFERENCES Ability(ability_id)

Species_Located_In(**pokedex**: INTEGER, **location_name**: VARCHAR(40));
- FOREIGN KEY (pokedex) REFERENCES Species(pokedex)
- FOREIGN KEY (location_name) REFERENCES Location(name)

Species_Belongs_To_Egg_Group(**pokedex**: INTEGER, **egg_group_id**: INTEGER)
- FOREIGN KEY (pokedex) REFERENCES Species(pokedex)
- FOREIGN KEY (egg_group_id) REFERENCES Egg_Group(egg_group_id)

Player_Owns_Badge(**trainer_id**: INTEGER, **badge_index**: INTEGER)

- FOREIGN KEY (trainer_id) REFERENCES Player(trainer_id)
- FOREIGN KEY (badge_index) REFERENCES Badge(badge_index)

5) Functional Dependencies:

Pokemon:
pokedex, pokemon_id → name
pokedex, pokemon_id → level
pokedex, pokemon_id → total_XP
pokedex, pokemon_id → nature
pokedex, pokemon_id → stat_increased
pokedex, pokemon_id → stat_decreased
pokedex, pokemon_id → HP_IV
pokedex, pokemon_id → attack_IV
pokedex, pokemon_id → defense_IV
pokedex, pokemon_id → speed_IV
pokedex, pokemon_id → ability_id
pokedex, pokemon_id → trainer_id
nature → stat_increased
nature → stat_decreased
total_XP → level

Ability:
ability_id → description
ability_id → name
name → ability_id
name → description

Species:
pokedex → name
pokedex → description
name → pokedex
name → description

Egg Group:
egg_group_id → name
egg_group_id → description
name → egg_group_id
name → description

Item:
name → description
name → price
name → rarity
name → location_name
price → rarity

Type:
name → colour
colour → name

Location:
name → climate
name → terrain_type

Trainer:
trainer_id → name
trainer_id → location_name

Gym Leader:
trainer_id → difficulty
trainer_id → cash_reward
trainer_id → specialty_type_name
trainer_id → badge_index
badge_index → trainer_id
badge_index → difficulty
badge_index → cash_reward
badge_index → specialty_type_name
difficulty → cash_reward
cash_reward → difficulty

Player:
trainer_id → money

Move:
move_id → name
move_id → power
move_id → pp
move_id → accuracy

move_id → description
move_id → type_name
name → move_id
name → power
name → pp
name → accuracy
name → description
name → type_name

Badge:
badge_index → name
name → badge_index

Species_Evolves_into:
old_pokedex, new_pokedex → evolution_level

6) Normalization:

The tables with non-trivial, non-key-related functional dependencies (thus requiring normalization) are the Pokemon, Gym Leader, and Item tables.

Pokemon:

Finding the minimal cover:
Step 1: Put FDs in standard form
- FDs are already in standard form

Step 2: Minimize LHS of each FD
- All FDs already have single-attribute LHS so no change to be made.

Step 3: Delete Redundant FDs
- Without “pokedex, pokemon_id → level”, “pokedex, pokemon_id → stat_increased”, and “pokedex, pokemon_id → stat_decreased”:
  - {pokedex, pokemon_id}+ = {name, level, total_XP, nature, stat_increased, HP_IV, attack_IV, defense_IV, speed_IV, ability_id, trainer_id}, which is the same closure as when “pokedex, pokemon_id → level” is included
  - So we can remove “pokedex, pokemon_id → level”, “pokedex, pokemon_id → stat_increased”, and “pokedex, pokemon_id → stat_decreased”
- We cannot remove any more FDs

Thus, the minimal cover is:
pokedex, pokemon_id → name
pokedex, pokemon_id → total_XP
pokedex, pokemon_id → nature
pokedex, pokemon_id → HP_IV
pokedex, pokemon_id → attack_IV
pokedex, pokemon_id → defense_IV
pokedex, pokemon_id → speed_IV
pokedex, pokemon_id → ability_id
pokedex, pokemon_id → trainer_id
nature → stat_increased
nature → stat_decreased
total_XP → level

Decomposition:
nature → stat_increased, stat_decreased violates 3NF, so not in 3NF. Decompose into:

- Pokemon_1(nature, stat_increased, stat_decreased)
- Pokemon_2(pokedex, pokemon_id, name, level, total_XP, nature, HP_IV, attack_IV, defense_IV, speed_IV, ability_id, trainer_id)

total_XP → level violates 3NF in Pokemon_2, so decompose into:

- Pokemon_3(total_XP, level)
- Pokemon_4(pokedex, pokemon_id, name, total_XP, nature, HP_IV, attack_IV, defense_IV, speed_IV, ability_id, trainer_id)

Final tables:
Pokemon_1(nature: VARCHAR(12), stat_increased: VARCHAR(12), stat_decreased: VARCHAR(12))
Pokemon_3(total_XP: INTEGER, level: INTEGER)
Pokemon_4(**pokedex**: INTEGER, pokemon_id: INTEGER, name: VARCHAR(20), **total_XP**: INTEGER, **nature**: VARCHAR(12), HP_IV:INTEGER, attack_IV: INTEGER, defense_IV: INTEGER, speed_IV: INTEGER, **ability_id**: INTEGER, **trainer_id**: INTEGER);
are now in 3NF

Gym Leader:

Finding the minimal cover:
Step 1: Put FDs in standard form

- FDs are already in standard form

Step 2: Minimize LHS of each FD

- All FDs already have single-attribute LHS so no change to be made.

Step 3: Delete Redundant FDs

- Without “trainer_id → cash_reward”:
  - {trainer_id}+ = {trainer_id, difficulty, cash_reward, specialty_type_name, badge_index}, which is the same closure as when “trainer_id → cash_reward” is included
  - So we can remove “trainer_id → cash_reward”
- Without “badge_index → difficulty”, “badge_index → cash_reward”, and “badge_index → specialty_type_name”:
  - {badge_index}+ = {trainer_id, difficulty, cash_reward, specialty_type_name, badge_index}, which is the same closure as when “badge_index → difficulty”, “badge_index → cash_reward”, and “badge_index → specialty_type_name” are included
  - So we can remove “badge_index → difficulty”, “badge_index → cash_reward”, and “badge_index → specialty_type_name”
- We cannot remove any more FDs

Thus, the minimal cover is:
trainer_id → difficulty
trainer_id → specialty_type_name
trainer_id → badge_index
badge_index → trainer_id
difficulty → cash_reward
cash_reward → difficulty

Decomposition:
“difficulty → cash_reward” violates 3NF, so we will decompose on “difficulty → cash_reward”:

- Gym_Leader_1(difficulty: VARCHAR(10), cash_reward: INTEGER)
- Gym_Leader_2(**trainer_id**: INTEGER, **difficulty**: VARCHAR(10), **specialty_type_name**: VARCHAR(9), **badge_index**: INTEGER)

There are no more violations of 3NF. Since all FDs are preserved, we don’t need to add any more relations.

Item:
This relation is in 2NF because all the non-key attributes depend on the whole primary key **name**. But, it is not in 3NF because “rarity” is transitively dependent on the primary key **name**, via “price”. This violates the 3NF condition.

Finding the minimal cover:
Step 1: Put FDs in standard form
name → description
name → price
name → rarity
name → location_name
price → rarity

Step 2: minimizing LHS of each FD

- All FDs already have single-attribute LHS so no change to be made.

Step 3: deleting the redundant FDs
We have name → price and price → rarity, so by transitivity, name → rarity is redundant.

Thus, we remove name → rarity.

And the minimal cover that we get is:
name → description
name → price
name → location_name
price → rarity

Identifying candidate keys:

{name}+ = {name, description, price, location_name, rarity}
{price}+ = {price, rarity}

Since {name}+ includes all the attributes, “name” is the candidate key.

Decomposition:

We will decompose Item(name, description, price, location_name, price, rarity) into:

1. Item_1(name: VARCHAR(255), description: VARCHAR(1000), **price**: INTEGER, **location_name**: VARCHAR(40))
    a. Primary key: name
    b. Candidate key: name
    c. Foreign key: location_name references Location(name)
2. Item_2(price: INTEGER, rarity: VARCHAR(10))

a. Primary key: price

Tables After Normalization:

Entities and One-to-Many-Relationships:

Pokemon_1(**pokedex**: INTEGER, pokemon_id: INTEGER, name: VARCHAR(20), **total_XP**: INTEGER, **nature**: VARCHAR(12), HP_IV: INTEGER, attack_IV: INTEGER, defense_IV: INTEGER, speed_IV: INTEGER, **ability_id**: INTEGER, **trainer_id**: INTEGER);

- trainer_id can be null
- FOREIGN KEY (pokedex) REFERENCES Species(pokedex)
- FOREIGN KEY (total_XP) REFERENCES Pokemon_3(total_XP)
- FOREIGN KEY (nature) REFERENCES Pokemon_2(nature)
- FOREIGN KEY (ability_id) REFERENCES Ability(ability_id)
- FOREIGN KEY (trainer_id) REFERENCES Trainer(trainer_id)

Pokemon_2(nature: VARCHAR(12), stat_increased: VARCHAR(12), stat_decreased: VARCHAR(12))

Pokemon_3(total_XP: INTEGER, level: INTEGER)

Ability(ability_id: INTEGER, name: VARCHAR(255), description: VARCHAR(1000));

- name is a candidate key

Species(pokedex: INTEGER, name: VARCHAR(12), description:VARCHAR(1000));

- name is a candidate key

Egg_Group(egg_group_id: INTEGER, name: VARCHAR(30), description: VARCHAR(1000));

- name is a candidate key

Item_1(name: VARCHAR(255), description: VARCHAR(1000), **price**: INTEGER, **location_name**: VARCHAR(40));

- price and location_name can be null
- FOREIGN KEY (price) REFERENCES Item_2(price)
- FOREIGN KEY (location_name) REFERENCES Location(name)

Item_2(price: INTEGER, rarity: VARCHAR(10))

- rarity cannot be null here because price cannot be null

Type(name: VARCHAR(9), colour: VARCHAR(50));

- colour is a candidate key

Location(name: VARCHAR(40), climate: VARCHAR(255), terrain_type: VARCHAR(255))

Trainer(trainer_id: INTEGER, name: VARCHAR(255), **location_name**: VARCHAR(40));

- location_name can be null
- FOREIGN KEY (location_name) REFERENCES Location(name)

Gym_Leader_1(**trainer_id**: INTEGER, **difficulty**: VARCHAR(10), **specialty_type_name**: VARCHAR(9), **badge_index**: INTEGER)

- badge_index is unique
- badge_index is a candidate key
- FOREIGN KEY (trainer_id) REFERENCES Trainer(trainer_id)
- FOREIGN KEY (difficulty) REFERENCES Gym_Leader_2(difficulty)
- FOREIGN KEY (specialty_type_name) REFERENCES Type(name)
- FOREIGN KEY (badge_index) REFERENCES Badge(badge_index)

Gym_Leader_2(difficulty: VARCHAR(10), cash_reward: INTEGER)

Player(**trainer_id**: INTEGER, money: INTEGER)

- FOREIGN KEY (trainer_id) REFERENCES Trainer(trainer_id)

Move(move_id: INTEGER, name: VARCHAR(50), power: INTEGER, pp: INTEGER, accuracy: INTEGER, description: VARCHAR(1000), **type_name**: VARCHAR(9))

- name is a candidate key
- FOREIGN KEY (type_name) REFERENCES Type(name)

Badge(badge_index: INTEGER, name: VARCHAR(15))

- name is a candidate key

Many-to-Many Relationships:

Species_Evolves_Into(**old_pokedex**: INTEGER, **new_pokedex**: INTEGER, evolution_level: INTEGER);

- evolution_level can be null
- FOREIGN KEY (old_pokedex) REFERENCES Species(pokedex)
- FOREIGN KEY (new_pokedex) REFERENCES Species(pokedex)

Pokemon_Has_Learned_Move(**pokedex**: INTEGER, **pokemon_id**: INTEGER, **move_id**: INTEGER)

- FOREIGN KEY (pokedex, pokemon_id) REFERENCES Pokemon(pokedex, pokemon_id)
- FOREIGN KEY (move_id) REFERENCES Move(move_id)

Species_Has_Type(**pokedex**: INTEGER, **type_name**: VARCHAR(9));

- FOREIGN KEY (pokedex) REFERENCES Species(pokedex)
- FOREIGN KEY (type_name) REFERENCES Type(name)

Species_Can_Learn_Move(**pokedex**: INTEGER, **move_id**: INTEGER)

- FOREIGN KEY (pokedex) REFERENCES Species(pokedex)
- FOREIGN KEY (move_id) REFERENCES Move(move_id)

Species_Can_Have_Ability(**pokedex**: INTEGER, **ability_id**: INTEGER);

- FOREIGN KEY (pokedex) REFERENCES Species(pokedex)
- FOREIGN KEY (ability_id) REFERENCES Ability(ability_id)

Species_Located_In(**pokedex**: INTEGER, **location_name**: VARCHAR(40));

- FOREIGN KEY (pokedex) REFERENCES Species(pokedex)
- FOREIGN KEY (location_name) REFERENCES Location(name)

Species_Belongs_To_Egg_Group(**pokedex**: INTEGER, **egg_group_id**: INTEGER)

- FOREIGN KEY (pokedex) REFERENCES Species(pokedex)
- FOREIGN KEY (egg_group_id) REFERENCES Egg_Group(egg_group_id)

Player_Owns_Badge(**trainer_id**: INTEGER, **badge_index**: INTEGER)

- FOREIGN KEY (trainer_id) REFERENCES Player(trainer_id)
- FOREIGN KEY (badge_index) REFERENCES Badge(badge_index)

7) SQL DDL Statements

Tables with no foreign keys, followed by normalization tables, followed by foreign key dependencies, followed by relationships

Entities and One-to-Many Relationships:

Tables with no Foreign Keys

a) Non-Normalized Tables

```
CREATE TABLE Location (
        name VARCHAR(40) PRIMARY KEY,
        climate VARCHAR(255) NOT NULL,
        terrain_type VARCHAR(255) NOT NULL
);

CREATE TABLE Type (
        name VARCHAR(9) PRIMARY KEY,
        colour VARCHAR(50) UNIQUE NOT NULL
);

CREATE TABLE Badge (
        badge_index INTEGER PRIMARY KEY,
        name VARCHAR(15) UNIQUE NOT NULL
);

CREATE TABLE Ability (
        ability_id INTEGER PRIMARY KEY,
        name VARCHAR(255) UNIQUE NOT NULL,
        description VARCHAR(1000) NOT NULL
);

CREATE TABLE Species (
        pokedex INTEGER PRIMARY KEY,
        name VARCHAR(12) UNIQUE NOT NULL,
        description VARCHAR(1000) NOT NULL
);

CREATE TABLE Egg_Group (
        egg_group_id INTEGER PRIMARY KEY,
```

```
    name VARCHAR(30) UNIQUE NOT NULL,
    description VARCHAR(1000) NOT NULL
);

CREATE TABLE Move (
    move_id INTEGER PRIMARY KEY,
    name VARCHAR(50) UNIQUE NOT NULL,
    power INTEGER NOT NULL,
    pp INTEGER NOT NULL,
    accuracy INTEGER NOT NULL,
    description VARCHAR(1000) NOT NULL,
    type_name VARCHAR(9) NOT NULL,
    FOREIGN KEY (type_name) REFERENCES Type(name)
);
```

b) Normalized Tables

```
CREATE TABLE Pokemon_2 (
    nature VARCHAR(12) PRIMARY KEY,
    stat_increased VARCHAR(12) NOT NULL,
    stat_decreased VARCHAR(12) NOT NULL
);

CREATE TABLE Pokemon_3 (
    total_XP INTEGER PRIMARY KEY,
    level INTEGER NOT NULL
);

CREATE TABLE Item_2 (
    price INTEGER PRIMARY KEY,
    rarity VARCHAR(10) NOT NULL
);

CREATE TABLE Gym_Leader_2 (
    difficulty VARCHAR(10) PRIMARY KEY,
    cash_reward INTEGER NOT NULL
);
```

Tables with Foreign Keys

```
CREATE TABLE Trainer (
        trainer_id INTEGER PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        location_name VARCHAR(40),
        FOREIGN KEY (location_name) REFERENCES Location(name)
);

CREATE TABLE Item_1 (
        name VARCHAR(255) PRIMARY KEY,
        description VARCHAR(1000) NOT NULL,
        price INTEGER,
        location_name VARCHAR(40),
        FOREIGN KEY (price) REFERENCES Item_2(price),
        FOREIGN KEY (location_name) REFERENCES Location(name)
);

CREATE TABLE Pokemon_1 (
        pokedex INTEGER NOT NULL,
        pokemon_id INTEGER NOT NULL,
        name VARCHAR(20) NOT NULL,
        total_XP INTEGER NOT NULL,
        nature VARCHAR(12) NOT NULL,
        HP_IV INTEGER NOT NULL,
        attack_IV INTEGER NOT NULL,
        defense_IV INTEGER NOT NULL,
        speed_IV INTEGER NOT NULL,
        ability_id INTEGER NOT NULL,
        trainer_id INTEGER,
        PRIMARY KEY (pokedex, pokemon_id),
        FOREIGN KEY (pokedex) REFERENCES Species(pokedex),
        FOREIGN KEY (total_XP) REFERENCES Pokemon_3(total_XP),
        FOREIGN KEY (nature) REFERENCES Pokemon_2(nature),
        FOREIGN KEY (ability_id) REFERENCES Ability(ability_id),
        FOREIGN KEY (trainer_id) REFERENCES Trainer(trainer_id)
);

CREATE TABLE Gym_Leader_1 (
        trainer_id INTEGER PRIMARY KEY,
        difficulty VARCHAR(10) NOT NULL,
        specialty_type_name VARCHAR(9) NOT NULL,
```

```
        badge_index INTEGER UNIQUE NOT NULL,
        FOREIGN KEY (trainer_id) REFERENCES Trainer(trainer_id),
        FOREIGN KEY (difficulty) REFERENCES Gym_Leader_2(difficulty),
        FOREIGN KEY (specialty_type_name) REFERENCES Type(name),
        FOREIGN KEY (badge_index) REFERENCES Badge(badge_index)
);

CREATE TABLE Player (
        trainer_id INTEGER PRIMARY KEY,
        money INTEGER NOT NULL,
        FOREIGN KEY (trainer_id) REFERENCES Trainer(trainer_id)
);
```

Many-to-Many Relationships:

```
CREATE TABLE Species_Evolves_into (
        old_pokedex INTEGER,
        new_pokedex INTEGER,
        evolution_level INTEGER,
        PRIMARY KEY (old_pokedex, new_pokedex),
        FOREIGN KEY (old_pokedex) REFERENCES Species(pokedex),
        FOREIGN KEY (new_pokedex) REFERENCES Species(pokedex)
);

CREATE TABLE Pokemon_Has_Learned_Move (
        pokedex INTEGER,
        pokemon_id INTEGER,
        move_id INTEGER,
        PRIMARY KEY (pokedex, pokemon_id, move_id),
        FOREIGN KEY (pokedex, pokemon_id) REFERENCES Pokemon (pokedex,
pokemon_id),
        FOREIGN KEY (move_id) REFERENCES Move(move_id)
);

CREATE TABLE Species_Has_Type (
        pokedex INTEGER,
        type_name VARCHAR(9),
        PRIMARY KEY (pokedex, type_name),
        FOREIGN KEY (pokedex) REFERENCES Species(pokedex),
        FOREIGN KEY (type_name) REFERENCES Type(name)
```

```
);

CREATE TABLE Species_Can_Learn_Move (
        pokedex INTEGER,
        move_id INTEGER,
        PRIMARY KEY (pokedex, move_id),
        FOREIGN KEY (pokedex) REFERENCES Species(pokedex),
        FOREIGN KEY (move_id) REFERENCES Move(move_id)
);

CREATE TABLE Species_Can_Have_Ability (
        pokedex INTEGER,
        ability_id INTEGER,
        PRIMARY KEY (pokedex, ability_id),
        FOREIGN KEY (pokedex) REFERENCES Species(pokedex),
        FOREIGN KEY (ability_id) REFERENCES Ability(ability_id)
);

CREATE TABLE Species_Located_In (
        pokedex INTEGER,
        location_name VARCHAR(40),
        PRIMARY KEY (pokedex, location_name),
        FOREIGN KEY (pokedex) REFERENCES Species(pokedex),
        FOREIGN KEY (location_name) REFERENCES Location(name)
);

CREATE TABLE Species_Belongs_To_Egg_Group (
        pokedex INTEGER,
        egg_group_id INTEGER,
        PRIMARY KEY (pokedex, egg_group_id),
        FOREIGN KEY (pokedex) REFERENCES Species(pokedex),
        FOREIGN KEY (egg_group_id) REFERENCES Egg_Group(egg_group_id)
);

CREATE TABLE Player_Owns_Badge (
        trainer_id INTEGER,
        badge_index INTEGER,
        PRIMARY KEY (trainer_id, badge_index),
        FOREIGN KEY (trainer_id) REFERENCES Player(trainer_id),
        FOREIGN KEY (badge_index) REFERENCES Badge(badge_index)
```

);

8) INSERT Statements:

Entities and One-to-Many Relationships:

Tables with no Foreign Keys

a) Non-normalized Tables

```
INSERT INTO Location (name, climate, terrain_type)
VALUES
        ('Pallet Town', 'Temperate', 'Grassland'),
        ('Viridian City', 'Temperate', 'Urban'),
        ('Pewter City', 'Mountainous', 'Rocky'),
        ('Cerulean City', 'Coastal', 'Beach'),
        ('Vermilion City', 'Coastal', 'Port');

INSERT INTO Type (name, colour)
VALUES
        ('Normal', 'Tan'),
        ('Fire', 'Red'),
        ('Water', 'Blue'),
        ('Electric', 'Yellow'),
        ('Grass', 'Green'),
        ('Ice', 'Light Blue'),
        ('Fighting', 'Brown'),
        ('Poison', 'Purple'),
        ('Psychic', 'Pink');

INSERT INTO Badge (badge_index, name)
VALUES
        (1, 'Boulder Badge'),
        (2, 'Cascade Badge'),
        (3, 'Thunder Badge'),
        (4, 'Rainbow Badge'),
        (5, 'Soul Badge'),
        (6, 'Marsh Badge'),
        (7, 'Volcano Badge'),
        (8, 'Earth Badge');
```

```
INSERT INTO Ability (ability_id, name, description)
VALUES
    (1, 'Intimidate', 'Lowers the foe''s Attack stat.'),
    (2, 'Static', 'Contact with the Pokémon may cause paralysis.'),
    (3, 'Levitate', 'Gives full immunity to all Ground-type moves.'),
    (4, 'Overgrow', 'Powers up Grass-type moves when the Pokémon''s HP is low.'),
    (5, 'Blaze', 'Powers up Fire-type moves when the Pokémon''s HP is low.'),
    (6, 'Torrent', 'Powers up Water-type moves when the Pokémon''s HP is low.'),
    (7, 'Guts', 'Boosts the Attack stat if the Pokémon has a status condition.');

INSERT INTO Species (pokedex, name, description)
VALUES
    (1, 'Bulbasaur', 'It is known to be extremely loyal, even after long-term
    abandonment. Bulbasaur can survive for days without eating. Its vines are long
    and strong enough to allow it to grab tree branches and pull itself up to reach
    berries.'),
    (4, 'Charmander', 'A fire burns at the tip of this Pokémons slender tail, which has
    blazed there since birth. The flame can indicate its health and mood, burning
    brightly when strong, weakly when exhausted, and blazing when enraged.'),
    (7, 'Squirtle', 'Its shell is a useful tool that it can withdraw into for protection or
    sleep. The grooved, rounded shape helps reduce water resistance, allowing it to
    swim at high speeds and spray foamy water with great accuracy.'),
    (25, 'Pikachu', 'Each cheek is a red circle that contains a pouch for electricity
    storage. It can use electricity to receive and send messages with other
    Electric-type Pokémon and has two horizontal brown stripes on its back.'),
    (63, 'Abra', 'Abra can sense danger through a telepathic radar and teleports to
    safety when it does. It sleeps 18 hours each day due to the strain of its telepathic
    powers and can teleport even while sleeping by hypnotizing itself.'),
    (92, 'Gastly', 'Its gaseous form makes it one of the lightest Pokémon in existence.
    It can phase through solid objects and form tangible hands from its gasses, but
    its body will dwindle away when exposed to strong winds.'),
    (147, 'Dratini', 'Its life energy is constantly building so it is always growing and
    can reach lengths of over six feet. It sheds its skin regularly, hiding behind rapid
    waterfalls during the process since the new skin is soft.');

INSERT INTO Egg_Group (egg_group_id, name, description)
VALUES
    (1, 'Mineral', 'Pokemon in this group are inorganic in nature'),
    (2, 'Amorphous', 'Pokemon in this group are amorphous, having no definite
    form'),
```

```
    (3, 'Grass', 'Pokemon in this group are plantlike in appearance'),
    (4, 'Water 3', 'Pokemon in this group resemble aquatic invertebrates'),
    (5, 'Water 2', 'Pokemon in this group are piscine (fishlike) in appearance'),
    (6, 'Water 1', 'Pokemon in this group are amphibious in nature'),
    (7, 'Bug', 'Pokemon in this group are insectoid (bug-like) in appearance'),
    (8, 'Dragon', 'Pokemon in this group are reptilian or draconic in appearance'),
    (9, 'Flying', 'Pokemon in this group are avian (birdlike) in appearance'),
    (10, 'Field', 'The largest group, Pokemon here are terrestrial in nature'),
    (11, 'Human-Like', 'Pokemon in this group are fully bipedal humanoids'),
    (12, 'Fairy', 'Pokemon in this group are petite and considered very cute'),
    (13, 'Monster', 'Pokemon in this group are saurian/kaiju-like in appearance and
    nature'),
    (14, 'Ditto', 'Ditto is the only Pokemon in this group, capable of breeding with
    most others'),
    (15, 'No Eggs Discovered', 'Pokemon in this group are unable to breed');

INSERT INTO Move(move_id, name, power, pp, accuracy, description, type_name)
VALUES
    (1, 'Tackle', 40, 35, 100, 'A physical attack in which the user charges and slams
into the target with its whole body.', 'Normal'),
    (2, 'Ember', 40, 25, 100, 'The target is attacked with small flames. May also leave
the target with a burn.', 'Fire'),
    (3, 'Water Gun', 40, 25, 100, 'The target is blasted with a forceful jet of water.',
'Water'),
    (4, 'Vine Whip', 45, 25, 100, 'The target is struck with slender, whip-like vines.',
'Grass'),
    (5, 'Thunder Shock', 40, 30, 100, 'A jolt of electricity is hurled at the target to
inflict damage. May also paralyze the target.', 'Electric'),
    (6, 'Confusion', 50, 25, 100, 'The target is hit by a weak telekinetic force. May
also leave the target confused.', 'Psychic'),
    (7, 'Rock Throw', 50, 15, 90, 'The user picks up and throws a small rock at the
target to attack.', 'Rock');
```

b) Normalized Tables

```
INSERT INTO Pokemon_2 (nature, stat_increased, stat_decreased) VALUES
    ('Adamant', 'attack', 'sp_attack'),
    ('Modest', 'sp_attack', 'attack'),
    ('Jolly', 'speed', 'sp_attack'),
    ('Timid', 'speed', 'attack'),
```

```
        ('Bold', 'defense', 'attack'),
        ('Impish', 'defense', 'sp_attack'),
        ('Calm', 'sp_defense', 'attack'),
        ('Careful', 'sp_defense', 'sp_attack'),
        ('Naive', 'speed', 'sp_defense'),
        ('Hasty', 'speed', 'defense'),
        ('Brave', 'attack', 'speed'),
        ('Quiet', 'sp_attack', 'speed'),
        ('Rash', 'sp_attack', 'sp_defense'),
        ('Lonely', 'attack', 'defense'),
        ('Mild', 'sp_attack', 'defense');

INSERT INTO Pokemon_3 (total_XP, level) VALUES
        (0, 1),
        (500, 10),
        (1000, 15),
        (1500, 18),
        (2000, 20),
        (2500, 22),
        (3000, 25),
        (3500, 28),
        (4000, 30),
        (4500, 33),
        (5000, 36),
        (5500, 39),
        (6000, 42),
        (6500, 45),
        (7000, 50);

INSERT INTO Item_2 (price, rarity)
VALUES
        (0, 'Quest'),
        (200, 'Common'),
        (600, 'Uncommon'),
        (1000, 'Rare'),
        (2500, 'Very Rare');

INSERT INTO Gym_Leader_2 (difficulty, cash_reward)
VALUES
        ('Easy', 1000),
```

('Medium', 2000),
('Hard', 3000),
('Very Hard', 5000),
('Expert', 10000);

Tables with Foreign Keys

INSERT INTO Trainer (trainer_id, name, location_name)
VALUES
(1, ‘Ash’, ‘Pallet Town’),
(2, 'Brock', 'Pewter City'),
(3, 'Misty', 'Cerulean City'),
(4, 'Lt. Surge', 'Viridian City'),
(5, 'Erika', 'Celadon City'),
(6, 'Gary', NULL);

INSERT INTO Item_1 (name, description, price, location_name)
VALUES
('Poké Ball', 'A device for catching wild Pokémon.', 200, 'Viridian City'),
('Potion', 'Restores the HP of a Pokémon by 20 points.', 200, 'Viridian City'),
('Super Potion', 'Restores the HP of a Pokémon by 50 points.', 600, 'Cerulean City'),
('TM28 - Dig', 'A TM that teaches the move Dig.', 1000, 'Celadon City'),
('Rare Candy', 'A candy that raises the level of a Pokémon by one.', 2500, NULL),
('Bicycle', 'A folding bicycle that is faster than the Running Shoes.', 0, 'Cerulean City');

INSERT INTO Pokemon_1 (pokedex, pokemon_id, name, total_XP, nature, HP_IV, attack_IV, defense_IV, speed_IV, ability_id, trainer_id)
VALUES
(6, 1, 'Charizard', 4000, 'Adamant', 31, 31, 20, 25, 1, 1),
(130, 1, 'Gyarados', 3500, 'Jolly', 25, 30, 18, 31, 2, 2),
(149, 1, 'Dragonite', 7000, 'Brave', 31, 31, 25, 10, 3, 3),
(68, 1, 'Machamp', 3000, 'Lonely', 28, 31, 15, 20, 4, 4),
(9, 1, 'Blastoise', 4500, 'Modest', 31, 10, 31, 20, 5, 5),
(65, 1, 'Alakazam', 6000, 'Timid', 20, 8, 15, 31, 6, 6),
(94, 1, 'Gengar', 5000, 'Mild', 22, 12, 18, 30, 7, 7),
(134, 1, 'Vaporeon', 2500, 'Quiet', 31, 15, 25, 12, 8, 8),

(135, 1, 'Jolteon', 5500, 'Rash', 25, 20, 18, 31, 9, 9),
(143, 1, 'Snorlax', 6500, 'Careful', 31, 25, 31, 5, 10, 10),
(242, 1, 'Blissey', 2000, 'Bold', 31, 5, 20, 15, 11, 11),
(248, 1, 'Tyranitar', 1500, 'Impish', 30, 28, 31, 18, 12, 12),
(350, 1, 'Milotic', 1000, 'Calm', 28, 15, 25, 20, 13, 13),
(395, 1, 'Empoleon', 0, 'Naive', 25, 20, 22, 28, 14, 14),
(448, 1, 'Lucario', 500, 'Hasty', 20, 28, 15, 31, 15, 15);

INSERT INTO Gym_Leader_1 (trainer_id, difficulty, specialty_type_name, badge_index)
VALUES
(2, 'Easy', 'Fighting', 1),
(3, 'Medium', 'Water', 2),
(4, 'Hard', 'Electric', 3),
(5, 'Very Hard', 'Grass', 4),
(6, ‘Expert’, ‘Normal’, 5);

INSERT INTO Player (trainer_id, money)
VALUES
(1, 5000),
(6, 15000)
(7, 0)
(8, 9)
(9, 10);

Many-to-Many Relationships:

INSERT INTO Species_Evolves_Into(old_pokedex, new_pokedex, evolution_level)
VALUES
(1, 2, 16),
(2, 3, 32),
(4, 5, 16),
(5, 6, 36),
(7, 8, 16)
(133, 134, NULL),
(133, 135, NULL),
(133, 136, NULL);

INSERT INTO Pokemon_Has_Learned_Move (pokedex, pokemon_id, move_id)
VALUES

```
    (25, 1, 1),
    (25, 1, 5),
    (1, 1, 4),
    (4, 1, 2),
    (7, 1, 3),
    (92, 1, 6);

INSERT INTO Species_Has_Type (pokedex, type_name)
VALUES
    (1, 'Grass'),
    (1, 'Poison'),
    (4, 'Fire'),
    (7, 'Water'),
    (25, 'Electric'),
    (63, 'Psychic'),
    (92, 'Ghost'),
    (92, 'Poison'),
    (147, 'Dragon');

INSERT INTO Species_Can_Learn_Move (pokedex, move_id)
VALUES
    (1, 1),
    (1, 4),
    (4, 1),
    (4, 2),
    (25, 1),
    (25, 5);

INSERT INTO Species_Can_Have_Ability (pokedex, ability_id)
VALUES
    (1, 4),
    (4, 5),
    (25, 2),
    (92, 3),
    (147, 1);

INSERT INTO Species_Located_In (pokedex, location_name)
VALUES
    (25, 'Viridian Forest'),
    (63, 'Cerulean City'),
```

```
    (92, 'Lavender Town'),
    (1, 'Pallet Town'),
    (4, 'Pewter City');


INSERT INTO Species_Belongs_To_Egg_Group (pokedex, egg_group_id)
VALUES
    (1, 1),
    (1, 2),
    (4, 1),
    (4, 6),
    (25, 4),
    (92, 5);


INSERT INTO Player_Owns_Badge (trainer_id, badge_index)
VALUES
    (1, 1),
    (1, 2)
    (1, 3)
    (1, 4)
    (2, 1)
    (2, 2),
    (3, 1);
```