Pokemon Trading Database System

1. Preamble

Last Modified: 4/17/2014
Team Members: Tanner Degenkolb, Carlos Diaz, Yuki Gonzalez, Gerjo-Ferey Tacaraya
Project Title: PokeBase
Summary: Pokemon trading database.
Idea Origin: Pokemon is a series of games where players catch and train virtual monsters to battle against other people or the computer AI. There are over 700 of these monsters to collect; the combination of statistics and moves available to each of these creatures is varied enough that a trading system between players should be implemented. Especially as we saw that there are no other ways of finding people to trade besides real life meetings.
Master Directory: ~/cdiaz3/PokeBase

2. Domain Description

The system will allow users to post pokemon they wish to trade along with all their important attributes and contact information. Users can also search the database for specific pokemon or attributes that they wish to obtain. We are also thinking of implementing a request system where people can post specific pokemon they want or have it appear when they log in if we implement a login system. Another feature we would like to implement along with the login system is to maintain each user's current inventory of Pokemon so that we can give recommendations based on party composition.

3. Use Case Scenarios

Pokemon Poacher/Farmer Ted:

Ted is only interested in catching Pokemon en-masse to trade for more and bigger Pokemon. He signs up for PokeBase and posts his entire inventory up for trade. He waits for someone to contact him and perhaps places more Pokemon on the market for trade to see if he can dupe someone else into trading him an even better Pokemon.

Pokemon Powerplayer Lane:

Lane is interested in min-maxing his party, so that he can be the best that ever was. He is looking for Pokemon built in very specific ways, because he doesn't want to waste his time training them. So, he signs up for PokeBase and does searches based on attributes such as Attack, Pokemon Type, and Level. If he doesn't find what he wants (which is most likely given his peculiar taste), he posts a request with his desired attributes.

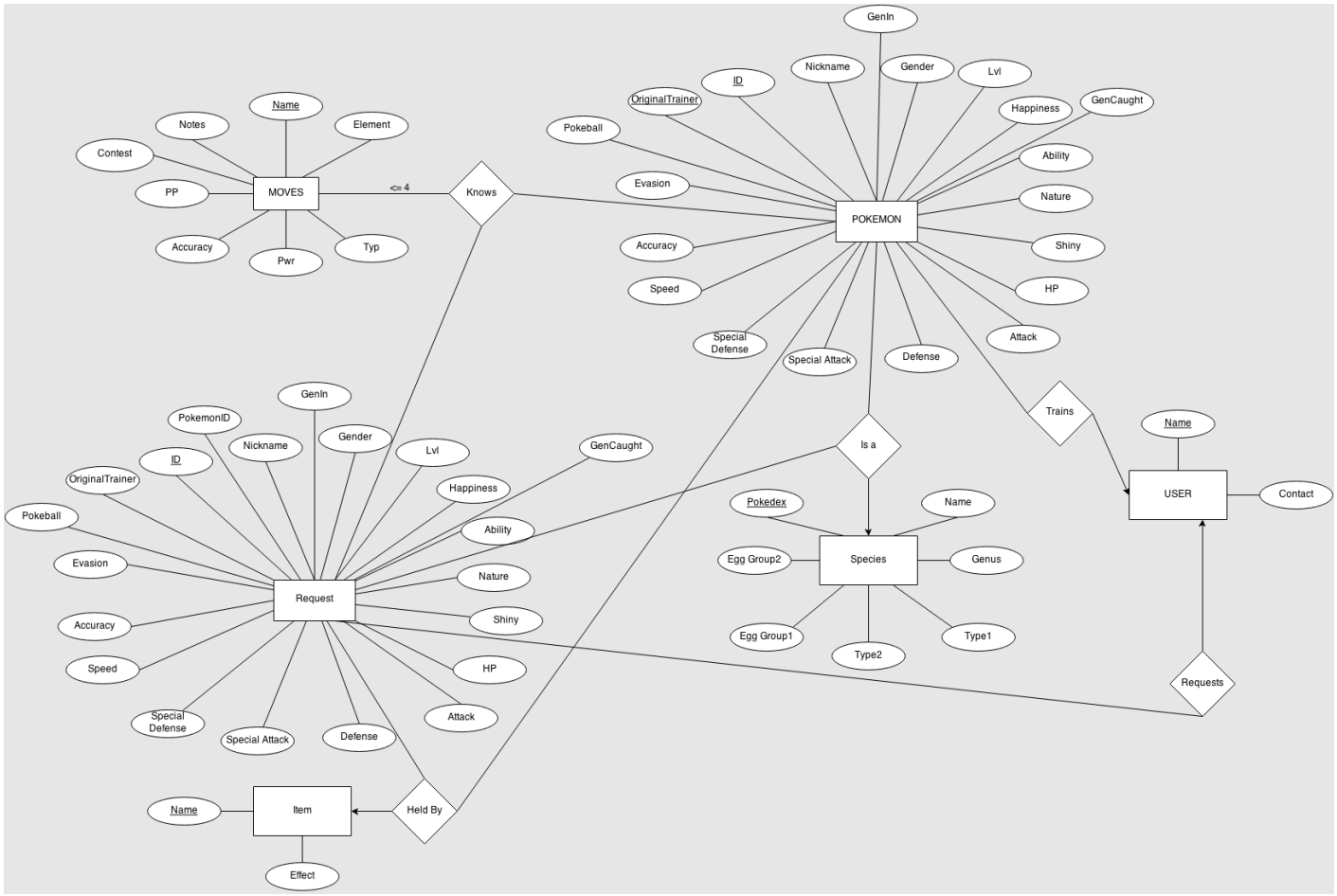
Pokenoob Chen:

Chen just started playing Pokemon and has no idea what he is doing. He signs up for PokeBase to browse available Pokemon and get an idea of what possibilities await him.

4. Sample Queries-English

a. A Mewtwo caught in generation 1 and is currently in generation 2
b. A pokemon who knows Surf and is part dark type
c. A pokemon that can mate with a bidoof and is in generation 3

5. Entity-Relationship Diagram



Explanation:

The POKEMON relationship represents all the game creatures that exist in our database; these are the things up for trade. Pokemon are distinguished by a unique Pokemon ID, which is assigned by the game system for each Pokemon. No other combination of attributes can yield a unique Pokemon; for instance, two distinct Pokemon can end up having the same moves,

species, trainer-assigned nickname, etc.

Pokemon can only know up to four moves, which is why we introduce a constraint in relation Knows; we also attempted to reflect this limitation in our schema for that relationship. Knows is many to many between POKEMON and MOVES because Pokemon learn multiple moves and some of the same moves may be learned by different Pokemon.

The Is A relation is many-to-one from POKEMON to Species. The reasoning here is that each Pokemon belongs to a particular “named-species”, and many Pokemon in the database will end up belonging to the same species. Species determines a Pokemon’s types (which moves it is weak to/strong against), the kinds of Pokemon it can mate with, etc. We use the Pokedex number as a key because that is what the game uses to uniquely catalogue species (as it turns out, species name is not a unique identifier for at least one case - the name Nidoran is shared by two distinct Pokemon).

The relation Trains is many-to-one from POKEMON to USER. A user owns/trains many Pokemon, but each Pokemon belongs to just one user. Currently, we only maintain a user’s contact information.

Held By is a many-to-one relation from POKEMON to Item. When trading Pokemon, the game allows each Pokemon involved to hold one item. Like Species, many Pokemon can end up holding the same type of item.

Requests is a new many-to-one relation from Request (another new entity) to USER. The idea is that each user can make multiple requests but each request belongs to a particular user (otherwise, how can we identify who to trade a requested Pokemon with?). For now, our approach for handling the requests system is to duplicate the POKEMON entity and its attributes as the Requests entity but to treat Requests as virtual, non-existing Pokemon. In addition, Requests has the same relationships with MOVES, Species, and Item as POKEMON. Requests require a unique, user assigned ID on top of the ID for POKEMON. Our thinking is that if a user specifies a PokemonID for a Pokemon that already exists in the database, he or she is really requesting that Pokemon and we may notify that user.

6. Relational Schema

Relations:
Pokemon(ID, nickname, gender, lvl, happiness, ability, nature, shiny, HP, attack, defense, special attack, special defense, speed, accuracy, evasion, originalTrainer, pokeball, genIn, genCaught, trainerName, pokedex)
Request(ID, pokemonID, nickname, gender, lvl, happiness, ability, nature, shiny, HP, attack, defense, special attack, special defense, speed, accuracy, evasion, originalTrainer, pokeball, genIn, genCaught, trainerName, pokedex)
Item(name, effect)

Moves(name, element, typ, pwr, accuracy, PP, contest)
User(name, address, contact)
Species(pokedex, name, genus, type1, type2, egg group1, egg group2)

Knows(pokemonID, requestID, originalTrainer, moveName1, moveName2, moveName3, moveName4)
HeldBy(pokemonID, requestID, originalTrainer, itemName)

7. Sample Queries-Relational Algebra

a.
R1 := $\sigma_{genCaught = 1 \text{ and } genIn = 2}((\pi_{pokedex}(\sigma_{name = \text{“Mewtwo”}}(Species))) \bowtie Pokemon)$

b.
R1 := $(\pi_{pokedex\#}(\sigma_{type1 = \text{“Dark”} \text{ or } type2 = \text{“Dark”}}(Species)) \bowtie Pokemon \bowtie (\sigma_{moveName1 = \text{“Surf”} \text{ or } moveName2 = \text{“Surf”} \text{ or } moveName3 = \text{“Surf”} \text{ or } moveName4 = \text{“Surf”}}(Knows))$

c.
R1 := $\sigma_{name = \text{“Bidoof”}}(Species)$
R2 := $\pi_{pokedex}(\sigma_{egg\ group\ 1 = R1.egg\ group1 \text{ or } egg\ group1 = R1.egg\ group2 \text{ or } egg\ group2 = R1.egg\ group1 \text{ or } egg\ group2 = R1.egg\ group2}(Species))$
R3 := $\sigma_{genIn = 3}$ (R2 $\bowtie$ Pokemon)

8. Functional Dependencies 1
User.name->contact
item.name->effect
Moves.name -> element, type, power, accuracy, PP, notes
pokedex -> name, genus, type1, type2, egg group1, egg group2
Pokemon.ID -> nickname, gender, level, happiness, ability, nature, shiny, HP, attack, defense, special attack, special defense, speed, accuracy, evasion, original trainer, BallIn, GenIn, GenCaught, trainerName, pokedex
Request.ID -> PokemonID, nickname, gender, level, happiness, ability, nature, shiny, HP, attack, defense, special attack, special defense, speed, accuracy, evasion, original trainer, BallIn, GenIn, GenCaught, trainerName, pokedex
Knows.pokemonID -> moveName1, moveName2, moveName3, moveName4
HeldBy.pokemonID, requestID -> itemName

All the multivalued dependencies we found have the same format as our functional dependencies (ie. key ->-> all other attributes).

9. Proof of Normal Form 1
We have no problematic multivalued dependencies on our domain and the left side of all

functional dependencies for each relation is a superkey (more precisely, the minimal key).

10-13 See Above

14. Database Implementation Status

Table Definitions:

```
CREATE TABLE pokemon (
 ID INTEGER, nickname CHAR(10),
 gender CHAR(1),
 lvl INTEGER,
 happiness INTEGER,
 ability CHAR(15),
 nature CHAR(15),
 shiny CHAR(1),
 HP INTEGER,
 attack INTEGER,
 defense INTEGER,
 specialAttack INTEGER,
 specialDefense INTEGER,
 speed INTEGER,
 accuracy INTEGER,
 evasion INTEGER,
 originalTrainer CHAR(10),
 pokeball CHAR(15),
 genIn INTEGER,
 genCaught INTEGER,
 trainerName CHAR(10),
 pokedex INTEGER,
 PRIMARY KEY (ID, originalTrainer)
);
```

1 tuple (a sample of what a user-owned Pokemon would look like)

```
CREATE TABLE request (
 ID INTEGER,
 pokemonID INTEGER,
 nickname CHAR(10),
 gender CHAR(1),
 lvl INTEGER,
 happiness INTEGER,
 ability CHAR(15),
 nature CHAR(15),
```

```
 shiny CHAR(1),
 HP INTEGER,
 attack INTEGER,
 defense INTEGER,
 specialAttack INTEGER,
 specialDefense INTEGER,
 speed INTEGER,
 accuracy INTEGER,
 evasion INTEGER,
 originalTrainer CHAR(10),
 pokeball CHAR(15),
 genIn INTEGER,
 genCaught INTEGER,
 trainerName CHAR(10),
 pokedex INTEGER,
 PRIMARY KEY (ID)
);
0 tuples

CREATE TABLE items (
 name CHAR(15),
 effect CHAR(255),
 PRIMARY KEY (name)
);
156 tuples

CREATE TABLE moves(
 name CHAR(15),
 typ CHAR(10),
 element CHAR(8),
 contest CHAR(6),
 PP INTEGER,
 pwr INTEGER,
 accuracy INTEGER,
 PRIMARY KEY (name)
);
617 tuples

CREATE TABLE users(
 name CHAR(32),
 address CHAR(255),
 contact CHAR(255),
 PRIMARY KEY (name)
```

```
);
0 tuples

CREATE TABLE species(
 pokedex INTEGER,
 name CHAR(15),
 genus CHAR(20),
 type1 CHAR(8),
 type2 CHAR(8),
 egg_group1 CHAR(12),
 egg_group2 CHAR(12),
 PRIMARY KEY (pokedex)
);
718 tuples

CREATE TABLE knows(
 pokemonID INTEGER,
 requestID INTEGER,
 originalTrainer CHAR(10),
 moveName1 CHAR(15),
 moveName2 CHAR(15),
 moveName3 CHAR(15),
 moveName4 CHAR(15)
);
0 tuples

CREATE TABLE heldby(
 pokemonID INTEGER,
 requestID INTEGER,
 originalTrainer CHAR(10),
 itemName CHAR(15)
);
0 tuples
```

URL’s:
betaweb.csug.rochester.edu/~/cdiaz3/Poke_Base/dbsetup-PokeTrader.sql
betaweb.cusg.rochester.edu/~/cdiaz3/Poke_Base/testqueries-PokeBase.sql

15. Data Abstraction Layer 1

We implemented a DAL for the relations pokemon and species.

URL:
betaweb.csug.rochester.edu/~/cdiaz3/Poke_Base/PokeDAL.php

16. Web Site DB Query Status 1

Currently, the index page performs the simple query “a” from section 4. In other words, it looks for a Mewtwo that is available for a particular set of generations/games. The page only displays relevant information, such as the Pokemon name, its original owner, egg groups, etc. This information will make it easy for a user to find the owner of the Pokemon and initiate a trade.

URL:
betaweb.csug.rochester.edu/~/cdiaz3/Poke_Base/index.php