Pokemon Trading Database System

1. Preamble

Last Modified: 5/9/2014
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: ~/tdegenko/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.

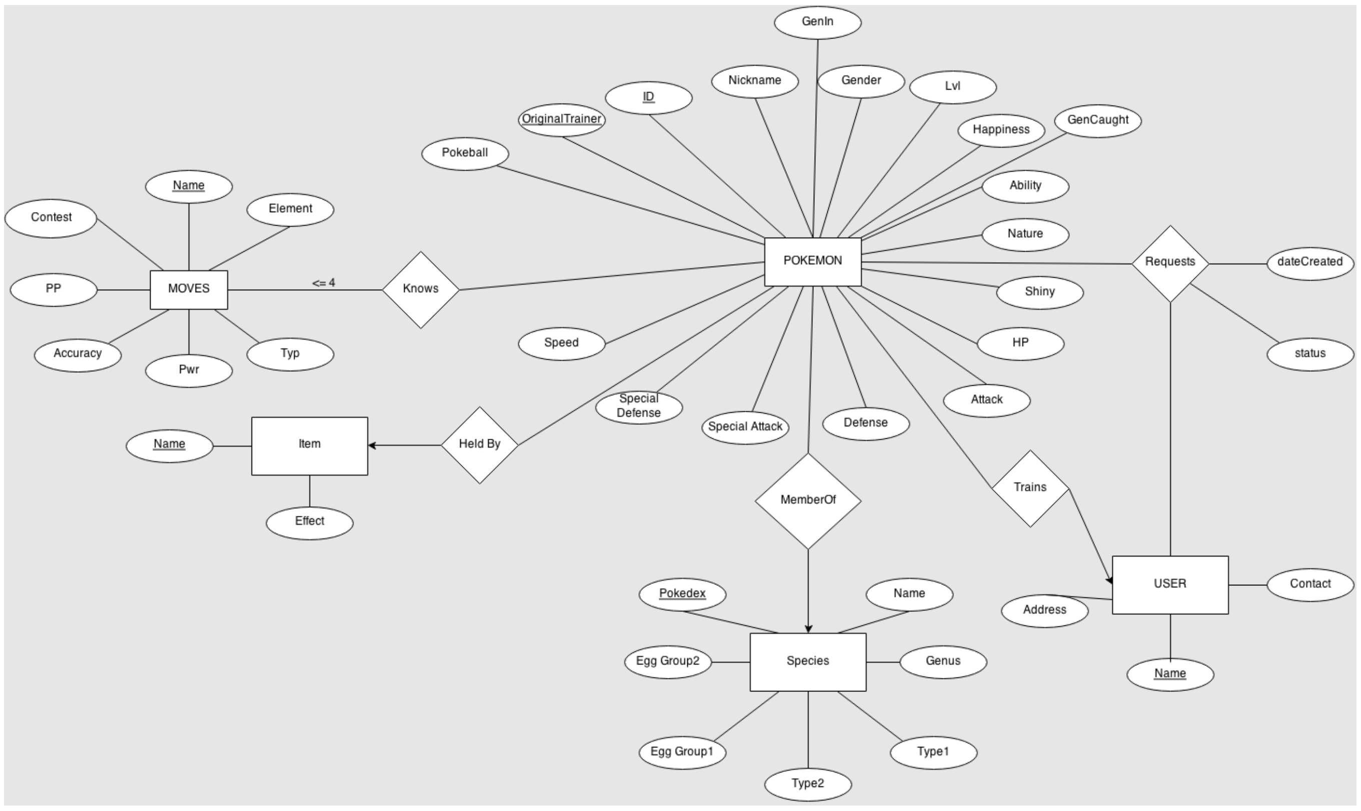
Pokenoob Chen:

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 entity set 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, and by their original trainer/owner. As it turns out, Pokemon ID's are not guaranteed to be unique across all games, so the game also embeds original trainer information to determine uniqueness during trades. The originial trainer's name is accessible to us, so we will use that along with Pokemon ID to identify particular 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 MemberOf 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. Users, or in game terminology “trainers”, are the people that trade and catch Pokemon. A user can own/train 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-many relation between POKEMON and USER. The idea is that each user can make multiple requests for different POKEMON and particular POKEMON can be requested by many USERS. We changed the implementation to reduce redundancies (for now, USERS may only request Pokemon that exist in the database). Each request has a creation date and its status (pending, successful, rejected).

6. Relational Schema

Before applying 4.5.3:
Pokemon(<u>ID</u>, nickname, gender, lvl, happiness, ability, nature, shiny, HP, attack, defense, special attack, special defense, speed, <u>originalTrainer</u>, pokeball, genIn, genCaught)
Item(<u>name</u>, effect)
Moves(<u>name</u>, element, typ, pwr, accuracy, PP, contest)
User(<u>name</u>, address, contact)
Species(<u>pokedex</u>, name, genus, type1, type2, egg group1, egg group2)

HeldBy(<u>ID</u>, <u>originalTrainer</u>, <u>itemName</u>)
Knows(<u>ID</u>, <u>originalTrainer</u>, moveName1, moveName2, moveName3, moveName4)
MemberOf(<u>ID</u>, <u>originalTrainer</u>, <u>pokedex</u>)

Requests(ID, originalTrainer, trainerName, dateCreated, status)
Trains(ID, originalTrainer, userName)

Final Schema:
Pokemon(ID, nickname, gender, lvl, happiness, ability, nature, shiny, HP, attack, defense, special attack, special defense, speed, originalTrainer, pokeball, genIn, genCaught, trainerName, pokedex, itemName)
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(ID, originalTrainer, moveName1, moveName2, moveName3, moveName4)
Requests(ID, originalTrainer, trainerName, dateCreated, status)

7. Sample Queries-Relational Algebra

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

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

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

8. Functional Dependencies 1

User.name->contact, address,
Pokemon.ID, originalTrainer -> nickname, gender, lvl, happiness, ability, nature, shiny, HP, attack, defense, special attack, special defense, speed, pokeball, genIn, genCaught, trainerName, pokedex, itemName.
Item.name -> effect

Moves.name -> element, typ, pwr, accuracy, PP, contest
Species.pokedex -> name, genus, type1, type2, egg group1, egg group2
Knows.ID, originalTrainer, moveName1-> moveName2, moveName3, moveName4
Requests.ID, originalTrainer, trainerName -> dateCreated, status

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,
 originalTrainer CHAR(10),
 pokeball CHAR(15),
 genIn INTEGER,
 genCaught INTEGER,
 trainerName CHAR(10),
 pokedex INTEGER,
 itemName CHAR(15),
 PRIMARY KEY (ID, originalTrainer),
 FOREIGN KEY (trainerName) REFERENCES users(name) ON DELETE CASCADE ON
UPDATE CASCADE,
 FOREIGN KEY (pokedex) REFERENCES species(pokedex) ON DELETE CASCADE,
 FOREIGN KEY (itemName) REFERENCES items(name) ON DELETE CASCADE,
) ENGINE=INNODB;
```

~9 tuples (samples for testing)

```
CREATE TABLE requests (
 ID INTEGER,
 originalTrainer CHAR(10),
 trainerName CHAR(10),
 dateCreated TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 status CHAR(32),
 PRIMARY KEY (ID, originalTrainer, trainerName),
	FOREIGN KEY (ID,originalTrainer) REFERENCES pokemon(ID,originalTrainer) ON
DELETE CASCADE,
 FOREIGN KEY (trainerName) REFERENCES users(name) ON DELETE CASCADE

) ENGINE=INNODB;
```

2 tuples (test)

```
CREATE TABLE items (
 name CHAR(15),
 effect CHAR(255),
 PRIMARY KEY (name)
) ENGINE=INNODB;
```

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)
) ENGINE=INNODB;
```

617 tuples

```
CREATE TABLE users(
 name CHAR(32),
 address CHAR(255),
 contact CHAR(255),
 passwd CHAR(225),
 PRIMARY KEY (name)
) ENGINE=INNODB;
```

2 tuples (testing login)

```
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)
) ENGINE=INNODB;
```

718 tuples

```
CREATE TABLE knows(
 ID INTEGER REFERENCES pokemon(ID),
 originalTrainer CHAR(10),
 moveName1 CHAR(15),
 moveName2 CHAR(15),
 moveName3 CHAR(15),
 moveName4 CHAR(15),
 PRIMARY KEY(ID, originalTrainer),
        FOREIGN KEY (ID,originalTrainer) REFERENCES pokemon(ID,originalTrainer) ON
DELETE CASCADE,
 FOREIGN KEY (moveName1) REFERENCES moves(name) ON DELETE CASCADE,
 FOREIGN KEY (moveName2) REFERENCES moves(name) ON DELETE CASCADE,
 FOREIGN KEY (moveName3) REFERENCES moves(name) ON DELETE CASCADE,
 FOREIGN KEY (moveName4) REFERENCES moves(name) ON DELETE CASCADE

) ENGINE=INNODB;
```

~9 tuples (samples for testing)

URL’s:
betaweb.csug.rochester.edu/~tdegenko/PokeBase/dbsetup-PokeTrader.sql
betaweb.csug.rochester.edu/~tdegenko/PokeBase/testqueries-PokeBase.sql
betaweb.csug.rochester.edu/~tdegenko/PokeBase/testqueries-PokeBase.txt

15. Data Abstraction Layer 1

We implemented a DAL for the relations pokemon and species.

URL:
betaweb.csug.rochester.edu/~tdegenko/PokeBase/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/~tdegenko/PokeBase/index.php

17. Data Abstraction Layer 2
DAL’s have been implemented for the rest of the relations on the database.

URLs:
betaweb.csug.rochester.edu/~tdegenko/PokeBase/PokeDAL.php
betaweb.csug.rochester.edu/~tdegenko/PokeBase/ItemDAL.php
betaweb.csug.rochester.edu/~tdegenko/PokeBase/MovesDAL.php
betaweb.csug.rochester.edu/~tdegenko/PokeBase/RequestsDAL.php
betaweb.csug.rochester.edu/~tdegenko/PokeBase/UserDAL.php

18. Web Site DB Query Status 2

This webpage is now rendered obsolete by the requests page and search page.
betaweb.csug.rochester.edu/~tdegenko/PokeBase/complex.php

This webpage contains a search form with fields for all attributes a pokemon might have and allows a user to search the database for a pokemon that matches all their criteria. The submit button links to a results page which displays a list of all matching pokemon. This webpage is the center of our website since the purpose of our project is for people to be able to find pokemon they want.
betaweb.csug.rochester.edu/~tdegenko/PokeBase/pokeSearch.php
19. Web Site DB Modification Status

This webpage contains a form with fields for all attributes a pokemon might have. It allows a user to insert a new pokemon into the database with all the specified attributes. An empty field will default the attribute to null for attributes that are not required. This is used to populate our database so that users can put their pokemon up for trade.
betaweb.csug.rochester.edu/~tdegenko/PokeBase/pokeAdd.php

This webpage contains a form with fields for all alterable attributes a pokemon might have. It lets a user change the information of their pokemon. It first searches for the pokemon with the same ID and originalTrainer, ensures that the pokemon owner is the user currently logged in, and then changes its attributes according to the inputs. This is used to allow users to reflect their

pokemon's growth while they remain untraded.
betaweb.csug.rochester.edu/~tdegenko/PokeBase/pokeEdit.php

20. Additional Database Capabilities

We have added foreign key constraints to all tables in the database. We intended to use CHECK syntax to enforce attribute constraints; however, we found that mySQL does not support CHECK, and any trigger we implemented to mimic CHECK on insert and update did not work. We did manage to make a scheduled event for removing old requests, which runs every 30 days and deletes requests older than 30 days.

21. Additional Website Capabilities

We have augmented our website with a login feature. Users now need to login using their unique username and their password before accessing PokeTrader. A request system is now in place; it allows users to add a request for a pokemon on the search results. Users can look at their requests as well as requests made of them, and can choose to reject or accept a request. All forms are automatically populated when appropriate and they enforce constraints for some attributes. Users can add, delete, and edit only their own pokemon for obvious security reasons; this is enforced by checking login information in every page (ie. cookies).

The main page is:
betaweb.csug.rochester.edu/~tdegenko/PokeBase/main.php

Once logged in, pages of interest would be:
betaweb.csug.rochester.edu/~tdegenko/PokeBase/pokeSearch.php
betaweb.csug.rochester.edu/~tdegenko/PokeBase/myPoke.php
betaweb.csug.rochester.edu/~tdegenko/PokeBase/pokeAdd.php
betaweb.csug.rochester.edu/~tdegenko/PokeBase/yourRequests.php
betaweb.csug.rochester.edu/~tdegenko/PokeBase/pokeEdit.php?ID=0&originalTrainer=Giovanni
.php

The easiest way to navigate around these pages is to use the top menu.
22. Project Demo

We will present a prezi presentation which demonstrates our database's and our website's features and layout. It is at
http://prezi.com/dhsntj9ecfjq/?utm_campaign=share&utm_medium=copy&rc=ex0share
We will also be giving a live demo to go along with the presentation. It will be hosted at
serenity.cif.rochester.edu/CSC296Final/main.php
for the duration of the presentation.