CS340 Introduction to Databases Final Project: BattleMagick!

By Ava Cordero

Project Outline:

The database we are designing for this project was originally inspired by and similar in nature to the universe created by Wizards of the Coasts in the popular trading card game, Magic the Gathering. In our implementation, we simplified the game and focused on the idea of wizards battling one another by drinking potions, casting spells, and befriending creatures which act on our bidding or even rally other creatures to assist the party as well, among other possibilities. The concept game is in its infancy, but when the database has been built, it can be used as the basis for a simple 2-dimensional video game. This has been a lot of fun to create, as well as very educational!

Database Outline:

Entities:
* Wizard – Each wizard has a name, special ability, life level, and magick level.
* Spells – Each spell has a name, cost, effect and a rate at which they can be cast. Spells have various positive and negative effects on wizards and creatures alike.
* Creatures – Each creature has a name, ability, attack, and defense. May aid a wizard in order to gain some traction in the BattleMagick Universe or, of course, vanquish enemy Wizards.
* Potions – Each potion has a name and effect. Only Wizards can drink potions, but be warned… Consume with caution!

Relationships:
* Wizards have potions and spells. These are many-to-many relationships, many wizards can have many potions and many spells. (2 relations described here)

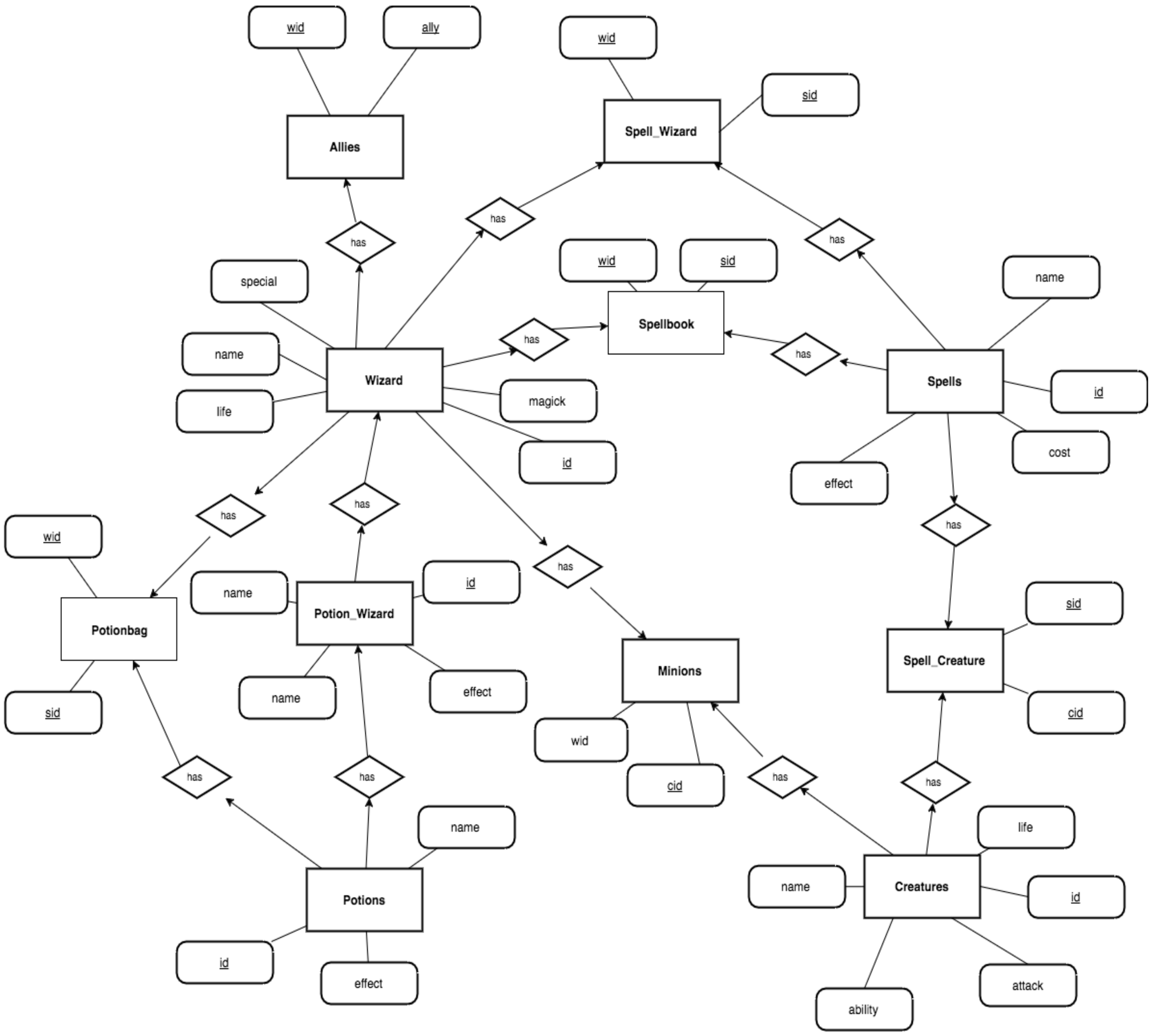
* Each Wizard has one (and only one) Special Spell. Special Spells may be cast for free!ff
* Wizards and creatures can be enchanted by potions and spells. These are also many to many relationships, many creatures can be enchanted by many spells. This is not the same as possessing a spell. (4 relations described here)

* Wizards can befriend creatures. This is a one-to-many relationship. One wizard can befriend many creatures, but a creature may only befriend one wizard.

* Wizards can form alliances with each other. This is a many-to-many relationship.

ER Diagram of Database:

The diagram below represents all of entities, relationships, constraints, and components of the database:



Database Schema:
The following diagram captures the attributes of each table. Additionally, it show each foreign key reference used in the database:

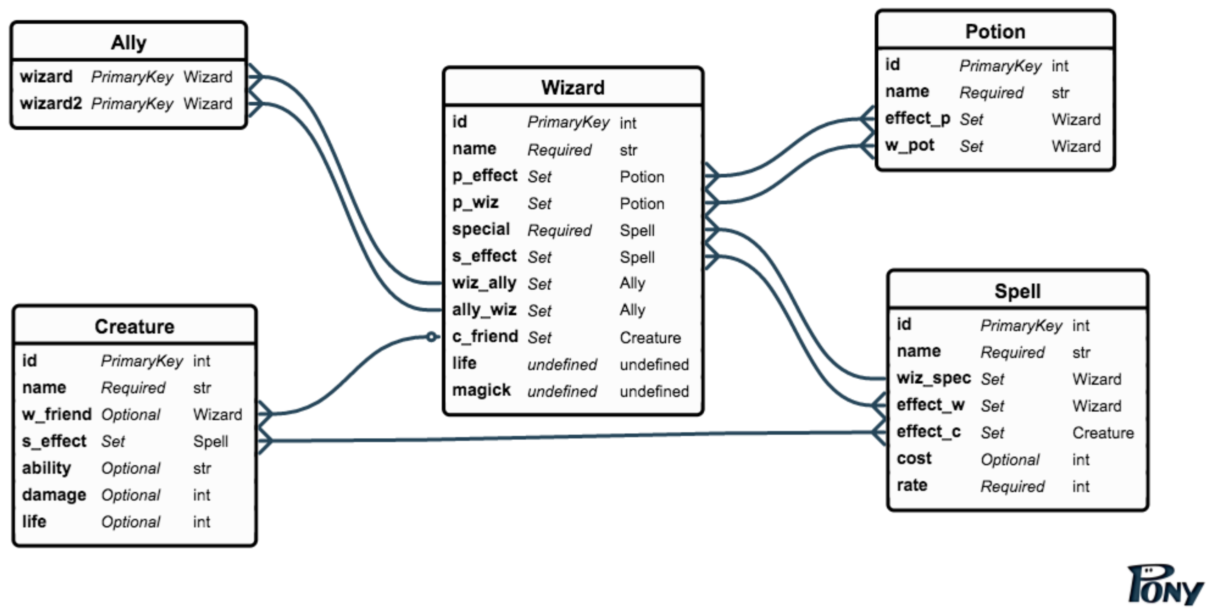


Table Creation Queries:
I want to see the queries your ran to build your tables. These should not be in any of the website code because you should not be dynamically building or deleting tables.

```
-- MySQL Schema for BattleMagick!
By Ava

CREATE TABLE `creature` (
  `id` int(11) PRIMARY KEY NOT
NULL AUTO_INCREMENT ,
  `name` varchar(30) NOT NULL,
  `w_friend` int(11) DEFAULT NULL,
  `ability` varchar(50) DEFAULT
NULL,
  `damage` int(11) DEFAULT NULL,
  `life` int(11) DEFAULT NULL,
  FOREIGN KEY (`w_friend`)
REFERENCES `wizard` (`id`) ON
DELETE CLEAR
) ENGINE=InnoDB DEFAULT
CHARSET=utf8;

CREATE TABLE `c_spell` (
  `creature` int(11) NOT NULL,
  `spell` int(11) NOT NULL,
  PRIMARY KEY
(`creature`,`spell`),
  KEY `idx_creature_spell`
(`spell`)
) ENGINE=InnoDB DEFAULT
CHARSET=utf8;

CREATE TABLE `potion` (
  `id` int(11) PRIMARY KEY NOT
NULL AUTO_INCREMENT,
  `name` varchar(30) NOT NULL
UNIQUE
) ENGINE=InnoDB DEFAULT
CHARSET=utf8;

CREATE TABLE `has_potion` (
  `potion` int(11) NOT NULL
PRIMARY KEY,
  `wizard` varchar(30) NOT NULL,
  KEY `idx_potion_wizard`
(`wizard`)
) ENGINE=InnoDB DEFAULT
CHARSET=utf8;

CREATE TABLE `potion_wizard_2` (
  `potion` int(11) NOT NULL,
  `wizard` varchar(30) NOT NULL,
  PRIMARY KEY (`potion`,`wizard`),
  KEY `idx_potion_wizard_2`
(`wizard`)
) ENGINE=InnoDB DEFAULT
CHARSET=utf8;

CREATE TABLE `spell` (
  `id` int(11) PRIMARY KEY NOT
NULL AUTO_INCREMENT,
  `name` varchar(30) NOT NULL
UNIQUE,
  `cost` int(11) DEFAULT NULL,
  `rate` smallint(6) NOT NULL
) ENGINE=InnoDB DEFAULT
CHARSET=utf8;

CREATE TABLE `spell_wizard` (
  `spell` int(11) NOT NULL,
  `wizard` varchar(30) NOT NULL,
  PRIMARY KEY (`spell`,`wizard`),
  KEY `idx_spell_wizard`
(`wizard`)
) ENGINE=InnoDB DEFAULT
CHARSET=utf8;

CREATE TABLE `wizard` (
  `id` int(11) PRIMARY KEY NOT
NULL AUTO_INCREMENT,
  `name` varchar(30) NOT NULL
UNIQUE,
  `special` int(11) NOT NULL,
  `life` int(11) NOT NULL,
  `magick` int(11) NOT NULL,
  FOREIGN KEY (`special`)
REFERENCES `spell` (`id`) ON
DELETE CLEAR,
  KEY `idx_wizard__special`
(`special`)
) ENGINE=InnoDB DEFAULT
CHARSET=utf8;
```

General Use Queries (30%)

I want to see all of the queries that will be used to select, update, add or delete data. Because many of these will be based on user input, use square brackets to act as place holders for variables that will be user provided. For example, if I were going to query based on employee salaries, I might have a query like this:
SELECT salary FROM employee WHERE salary > [salaryInput]; Another example
INSERT INTO employee(name, age) VALUES ([user],[name]);

A JavaScript function that will insert into the given table the provided data, and will render the new table to the browser:

```
app.post('/insert',function(req,re
s,next){
    if (req.body.table ===
'wizard'){
        attribs="
(`name`,`special`,`life`,`magick`)
"+
            " VALUES ((?),(?),(?),
(?))";
        fields =
[req.body.name,req.body.special,re
q.body.life, req.body.magick];
    } else if (req.body.table ===
'potion'){
        attribs=" (`name`) VALUES
(?)";
        fields = [req.body.name];
    } else if (req.body.table ===
'creature'){
        attribs="
(`name`,`w_friend`,`ability`,`dama
ge`,`life`)"+
            " VALUES ((?),(?),(?),
(?),(?))";
        fields
=[req.body.name,req.body.w_friend,
req.body.ability,req.body.damage,
req.body.life]
    } else if (req.body.table ===
'spell'){
        attribs="
(`name`,`cost`,`rate`) VALUES
((?),(?),(?))";
        fields =
[req.body.name,req.body.cost,req.b
ody.rate]
    }
    var context = {};
    console.log=(req.body)
```

```
        mysql.pool.query("INSERT
INTO "+req.body.table+attribs,
fields, function(err, result){
            if(err){
                next(err);
                return;
            }
        mysql.pool.query('SELECT *
FROM '+req.body.table,
function(err, rows, fields){
        if(err){
            next(err);
                return;
        }
        context.fields = fields;
        context.results = rows;
        context.table =
req.body.table;
        res.render('table',
context);
       });
    });
});
```

General JavaScript function to delete items from a table and render the new table to the browser:

```
app.post('/delete',
function(req,res,next){
  var context = {};
    console.log("table returned
is: ",req.body.table)
  mysql.pool.query("DELETE FROM
"+req.body.table+" WHERE
`id`=(?)", parseInt(req.body.id),
function(err, result){
    if(err){
      next(err);
      return;
    }
    mysql.pool.query('SELECT *
FROM '+req.body.table,
function(err, rows, fields){
    if(err){
        next(err);
        return;
    }
    context.fields = fields;
    context.results = rows;
    context.table =
req.body.table;
    res.render('table', context);
    });
  });
});
```

I hope you enjoy the website! ~Ava