Magic the Gathering Database Project Report

Carlos A. Rios

Derek Jones

April 29, 2017

Abstract

Magic the Gathering is a popular trading-card game with over 30,000 cards available since of April 29, 2017. We took upon the task of creating a database capable of holding all these cards in MySQL, and allowing the user to be able to search cards by specific search fields and allow users to create decks that are stored into the database. The design and implementation of the database was created and revised so that each relation in our schema was decompose into Boyce-Codd Normal Form (BCNF). Designing the schema was a complicated task as it required a large amount of decomposition of information, such as separating SetName from SetCode to allow BCNF from the entity MTGSet and so forth. The vast amount of information found in the Magic the Gathering rules, card, color, color identity, types, set, and formats was a complicated task. But nonetheless, we were able to resolve this problem by creating a relational model that supports BCNF and allows user to be able to search from the database to find a card(s) that meet specific requirements provided by them and allow construction of decks by user based on format.

Table of Contents

[Introduction 1](#__RefHeading___Toc1119_57376967)

[Product Requirements 3](#__RefHeading___Toc377_1431657501)

[Design-Part 1: E/R Diagram, Relational Model, and BCNF Verification 5](#__RefHeading___Toc387_1431657501)

[References 6](#__RefHeading___Toc385_1431657501)

# Introduction

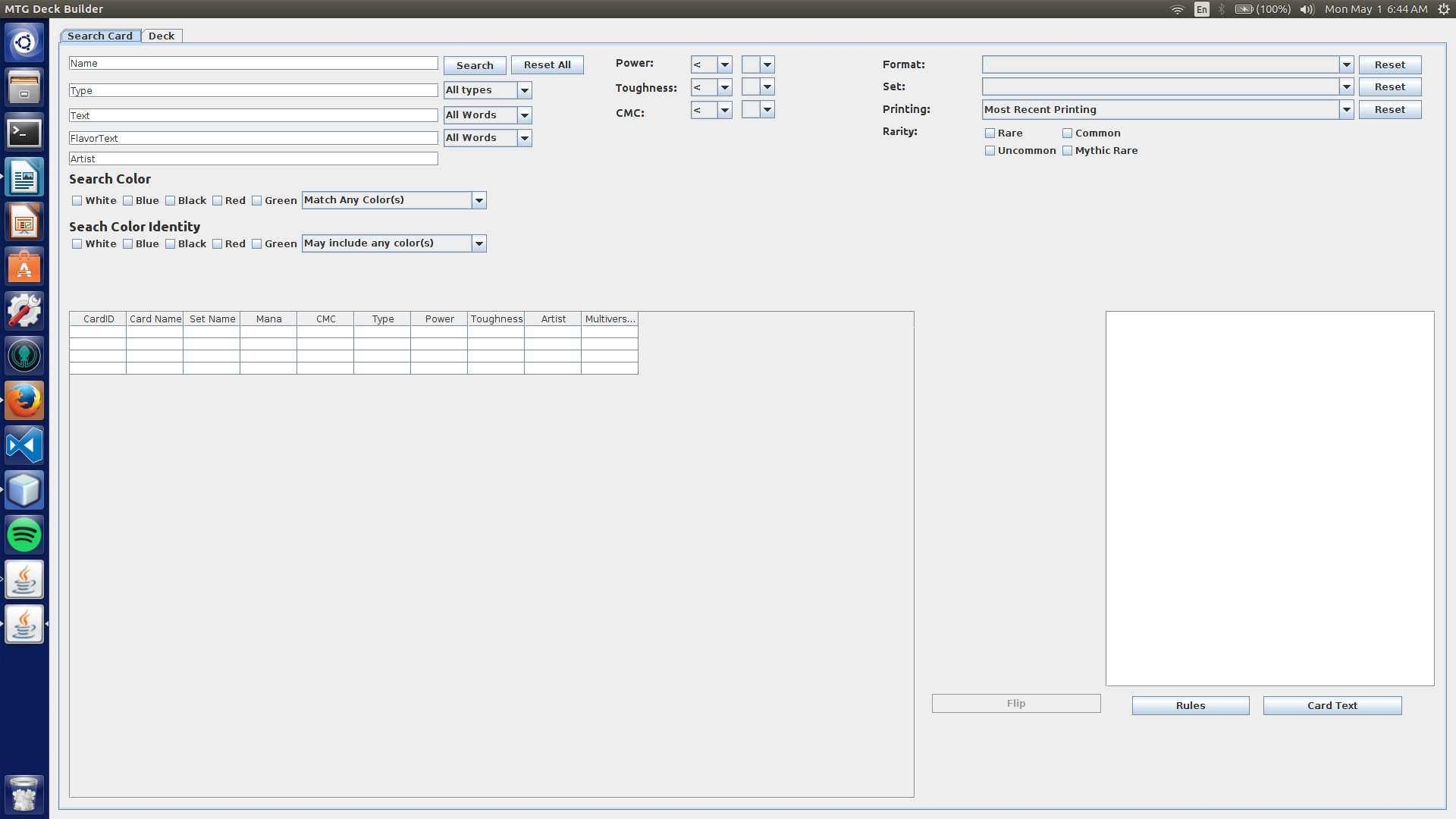
Our project for Database Management System involved creating a small real-world database application and we decided to create a database for the trading-card game “Magic the Gathering” using Java and MySQL. Magic the Gathering has over 30,000 card released, as of May 2017, and with such an immense collection of cards, it would be ideal to create a database that stores all the cards and a application that allows users to pull from the database based on search fields. Each individual card has the following information on them:

* Card Name (and the names of other card if layout is not standard)
* Mana Cost
* Converted Mana Cost (CMC)
* Colors
* Color Identity
* Types (includes Super Type, Type, and Sub Type)
* Text (includes Card Text, Flavor Text, and Ruling Text)
* Rarity
* Power
* Toughness
* Multiverse ID
* Artist
* Legalities
* Layout
* Loyalty
* Set Name

Though, there are more fields and information to an individual card, we decided that these were the necessities in order to create a database. With the help of MTGJson.com, we were able to obtain the JSON file, version 3.8.3, that included the most recent set, Amonkhet, that includes every set information and card information from those sets.[1] Every set includes the following information:

* Set Name
* Set Code
* Released Date
* Block
* Set Type

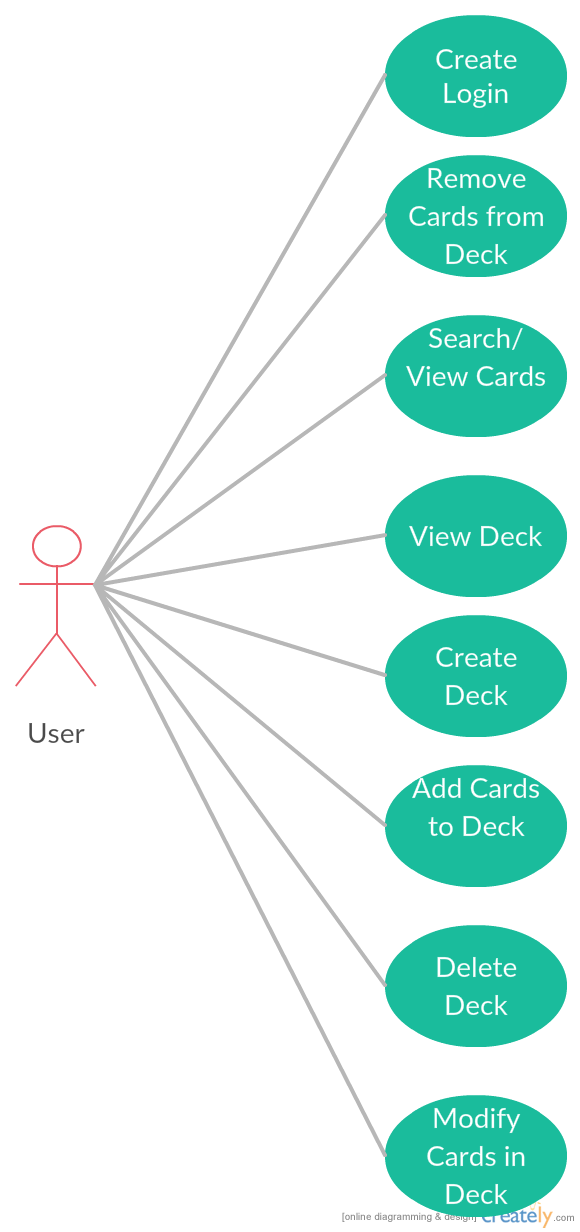
With the provided JSON file and Google’s GSON, a JAR File with a library that can convert JSON into Java Object[2], we were able to parse the file and import the information into our database. To allow user an ease of access to the database, we also created a GUI, as seen below in Figure 1, with search fields that can be edit to fit the description of a card(s) that a user is looking for.

Figure 1 - MTG Search GUI in Ubuntu 16.04  


The process of creating the database proved difficult as the number of information and the normalization of such information challenged us with many design problem. In the next section, we will begin by introducing the product requirements, which list all actors and use cases by those actors. This will help us identity what it is that we will need in order to design the relational model and entity relationship (ER) diagram. By doing so, we can showcase what kind of problem we faced in designing such database and our approach to solve the problem.

# Product Requirements

The actor, shown below Figure 2, is the User. In order to become a User of the system, they must first create a login account, which consist only of username and password. Afterwards, the user can do the following:

Figure 2 - Use Cases

* Create New Login
* Search/View Cards
* View Deck
* Create Deck
* Delete Deck
* Add Cards to the Deck
* Remove Cards from the Deck
* Update Qty of Cards in the Deck

User can search/view cards based on the specified search fields, which are the following:

* Card Names
* Card Type
* Artist
* Card Text
* Flavor Text
* Color (Color and/or ColorIdentity)
* Cards Stats (CMC, Toughness, Power)
* Rarity
* Format
* Set
* Printing

All these fields narrow down the search result based only on what fields have been changed. Card name refers to the name of the card being searched for. Card type refers to type of the card, an example being a card that includes the type “Goblin” or “Legendary Creature Wizard”. Artist refers to the person who drew that card. Card text is the text on the card that is not a favor text and can be seen as what this can do if it has any special abilities. Flavor text refers to the lore/story of the card through some quote or description that is italicized. Color is simply the color of the card, usually based on the color symbol on the top right of the card. Card Stats are the cost it takes to cast the card and the power and toughness of the creature. Rarity refers to the ability of attaining the card from the set with

common being the easiest to pull from a pack, followed by uncommon, then rare, and last is mythic rare being the rarest. Format refers to the format in which this card is legal or illegal to play in. Set refers to the set in which the card came from. The printing option refers to the printing of the card, which can be the original, reprints, newest card, or all the printings so far. And example of a card can be seen below as reference in Figure 3.

Figure 3 – MTG Card

Image from Wizards of the Coast Gatherer website.[3]

Figure 3 - MTG Card  


Image from Wizards of the Coast Gatherer website. [3]

View Deck cases provides us a way for users to be able to view their user created deck, if there any, which is an important component to deck building. Creating deck also helps user achieve this by allowing creation of the deck based on the Format they wish to construct in. Deleting deck helps user remove any unwanted decks they might have created. Adding card to the deck allows user to construct their decks by adding cards to it. Deleting card from the deck allows the removal of unwanted cards from the deck. And lastly, we have the ability to update the quantity of each card in the deck’s mainboard and sideboard.

These use cases define the user actor in the database and in our application. These use cases grant the user the ability to search through the Magic the Gathering database filled with cards and allow them the ability to create a deck that is stored in the database. Currently, the only way to update the database is through running a Java file that parses the JSON file and inserts every information into their corresponding entities. The next section, we will go discuss E/R diagram, relational model, and BCNF verification of the database.

# Design-Part 1: E/R Diagram, Relational Model, and BCNF Verification

# References

[1]. MTGJson. <https://mtgjson.com/>

[2]. GitHub Google GSON. <https://github.com/google/gson>

[3] . Wizards of the Coast. <http://gatherer.wizards.com/Handlers/Image.ashx?multiverseid=407644&type=card>