**Phase 2 – Formal Specification with a Relational Model**

**1. Complete E/R Diagram**

-The complete E/R diagram is separate from the word document as it does not fit on the word document to be able to see clearly due to resolution. But there are currently 11 entities (3 of them being a part of a “is-a” relationship). The following entities and their keys are the following:

1. User
   * Primary Key: Username
2. Deck
   * Primary Key: DeckID
   * Foreign Key: Format (FormatName from Format entity)
3. User Deck (“is-a” relationship with Deck entity)
   * Primary Key: fk\_Username & fk\_DeckID (Combined)
   * Foreign Key: fk\_Username & fk\_DeckID (both from User and Deck entity respectively)
4. Tournament Deck (“is-a” relationship with Deck entity)
   * Primary Key: PlayerID & fk\_DeckID (combined)
   * Foreign Key: fk\_DeckID (Deck entity)
5. Set
   * Primary Key: SetName
6. Card
   * Primary Key: CardID
7. Format
   * Primary Key: FormatName
8. Ruling
   * Primary Key: fk\_CardID & TextRuling (Combined)
   * Foreign Key: fk\_CardID (Card entity)
9. Type
   * Primary Key: typeID

**(10)** Color

* Primary Key: colorID

**(11)** ColorIdentity

* Primary Key: colorID

**(12)** Split\_Flip Card (“is-a” relationship with Card entity)

* PrimaryKey: NamesOnCard & fk\_CardID (Combined)
* Foreign Key: fk\_CardID (Card entity)

I will explain each of the entities and their relationship with one another starting off with Card entity. Card entity has a relationship with Set entity because each card is unique to the set, so a card belongs to 1 set only and a set has from 1 to many cards in it. In other words, every set has cards and each card is in 1 set.

Card entity has a relationship with Format entity because each card is can be allowed, banned, or restricted in each of the formats (currently 34 formats) and each format can include atleast 1 to many card in it. A format is a game type in which the set of rules differ, thus allowing or disallowing certain cards into the format, which is why the relationship is the way it is.

The separation of Type from Card (originally type was an attribute of Card) was due to cards having multiple types. It is to note that CardType attribute in Card is different from the Type entity as Type entity was formed from multi-attributed columns found in Card (“supertypes”, “types”, and “subtypes”) while CardType is a single attribute column, like full name or something. Each of the multi-attributed type can have 1 to many instances in a single card. For example, a “types” can include 1 to many instances in a single card. The same for “subtypes” and “supertypes”. So, we separated multi-attribute columns into their own table due to criticism from other developers against having multi attributed columns because they are “bad designs” in practice. Therefore, Type entity includes the type id, type name, and the type of type name (“types”, “supertypes”, or “subtypes”) and we added a relationship table to connect Card entity and Table entity with their respected type.

The same was done with Split\_Flip Card entity but it was done with a “is-a” relationship with Card entity as Split/Flip/Merge Card are still Cards, but rather more of a subset of Cards. Another reason this was done as there is a relationship between the Card and the names on that Card as split cards can have 2 or more cards related with a single card and flip cards have two different names on the card. So, we needed a way to connect each card name to a card and decided this a way to show that relationship between the card and names of that card by having a CardID (Foreign Key from Card entity) in Split\_Flip Card entity and the name of the card as another column. So, each row shows the connection between the card and the names on that card in Split\_Flip Card entity.

The Card entity and Ruling entity relationship was done in a way to show that each card has 0 to many rulings and that each of these ruling is connect to only 1 card. So, we decided to have cardID as a foreign key in Ruling entity to show the rules associated with that card.

Color entity and ColorIdentity entity both share the attributes (though I’m unsure whether this is a correct implementation) but they both differ because a Card could have a special rule that might render its Card\_Color relationship as colorless (which is nonexistent on the relationship as it is null) but could still potential have a Color Identity due to the symbols on the card. Color Identity does not care for special rulings, but rather for the identity of the colors which the card represents. The perfect example of this is of the card “Ghostfire”. Ghostfire’s color is colorless but the color identity is red. Color of the card does not take into consideration what is on the text side of the color, only the mana cost and special rulings. Therefore, it was a must to separate these two, even though they both shared the same columns. But both share the same multiplicity as a color or color identity can be in 1 to many cards and a card may contain 0 to 5 color or color identity to it.

User entity and Card entity share a relationship because a User interacts with the Card entity by viewing the information associated with it and card are viewed/accessed by the User. A user can view 0 to many cards, while a card can be viewed by 0 to many users.

A Deck entity and Card entity share a relationship because every deck includes 0 to many cards and a card may be in 0 to 4 decks. There can be at most 4 of the same card in each deck, with the exception of the basic lands. But other than, a deck can contain 0 as the creation of it does not have to include a card. And a card might not be necessarily in a deck due to lack of playability.

The userDeck entity and TournamentDeck entity share a “is-a” relationship with as both as different enough to related with Deck entity but different as TournamentDecks might not neccesarily be users within the system. UserDeck pertains to the creation of decks by user of this system and tournament decks are decks that won tournaments, usually represented as the top 8 decks of the tournament. We wanted to add tournament decks to the list for users to see and potentially replicate in their own deck creation. These decks will be added by Administrator.

The Deck entity and User entity share a relationship as User can create 0 to many decks and each deck is the creation of 1 user. Administrators are in charged of inputting the tournament decks while users are responsible for creating their user decks. In others words, a deck has only 1 user associated with it and a user can create many decks.

The relationship between User entity and Set entity exists solely for the administrator to add set

and manage the set, which each admin can manage 0 to many sets, while a set is managed by atleast 1 user(admin) to many. The user (admin) also supervises the other users of the system.

**2. Relational Model**

User (Username, password, role)

Deck (DeckID, DeckName, fk\_Format)

UserDeck(DeckID, Username, Visible)

TournamentDeck(PlayerID, DeckID, PlayerName)

Card(CardID*,* CardName, Layout, CardText, ManaCost, MultiverseID, Toughness, Power, Rarity, FlavorText, CardType, Loyalty, SetName, CMC, Artist)

Ruling(CardID,RulingText, RulingYear)

Split\_Flip\_Card(NamesOnCard, CardID)

Color(ColorID, ColorSymbol, Color)

ColorIdentity(ColorID, ColorSymbol, Color)

Type(TypeID, Type, TypeName)

Format(FormatName)

Set(SetName, Code, SetType, ReleasedDate, Block)

Format\_Card(FormatName, CardID, BanType)

Card\_Color(CardID, TypeID)

Card\_ColorIdentity(CardID, TypeID)

Deck\_Card(DeckID, CardID, MainboardQty, Sideboard Qty, CardName)

**Functional Dependencies**

userUsername → userPassword, userRole

deckDeckID → deckDeckName, deckFormat

deckDeckID, userUsername → userdeckVisible

tournamentdeckPlayerID, tournamentdeckDeckID → tournamentdeckPlayerName

cardCardID → cardCardName, cardLayout, cardCardText, cardManaCost, cardMultiverseID, cardToughtness, cardPower, cardRarity, cardFlavorText, cardCardType, cardLoyalty, setSetName, cardCMC, cardArtists

cardCardID, rulingRulingText → rulingRulingYear

colorColorID → colorColorSymbol, colorColor

coloridentityColorID → coloridentityColorSymbol, coloridentityColor

typeTypeID → typeType, typeTypeName

setSetName → setCode, setSetType, setReleasedDate, setBlock

formatFormatName, cardCardID → format\_cardBanType

deckDeckID, cardCardID→ deck\_cardMainboardQty, deck\_cardSideboardQty

All of our relations are in BNCF.

**4. Database Prototype**

The tools that we are using in our project are Netbeans for the Java JFrame GUI, GitHub for version control and allowing exchanging of information, ERDPlus.com for the creation of the E/R diagram, and MTGJson.com’s JSON Set file for the information of every card and set in the trading-card game “Magic the Gathering”.

We populated the SQL Database with the use of GSON to read through the JSON file and gather the information from. Then, we use java.sql to store the information parsed from the JSON into the SQL Database. Java file for the implementation will be provided.

**5. Project Time Table**

Major Tasks to be complete:

1. Finish Implementing Client GUI
2. Finish Implementing Admin GUI
3. Implement Card Searching Database functions to Admin and Client GUI’s
4. Implement Deck Build Functions to Client GUI
5. Implement Managing Set Functions to Admin GUI
6. Implement Managing Card Functions to Admin GUI
7. Implement Managing Format Functions to Admin GUI
8. Implement Managing Type Functions to Admin GUI
9. Implement Managing Ruling Functions to Admin GUI
10. Implement Managing User Functions to Admin GUI
11. Implement Managing Deck Functions to Admin GUI
12. Write Final Report

The following will use the previous table’s number and assigned each of the task to be completed by the following person:

**Carlos Rios:**

* Tasks 1, 2, 3, and 4 to be done by March 31, 2017
* Tasks 5 and 6 to be done by April 15

**Derek Jones:**

* Tasks 7, 8, 9 to be done by March 31
* Task 10, 11 to be done by April 15

For Tasks 11, we will break that up accordingly. So, for the moment Tasks 11 will be assigned to the both of us, each in charge of a specific part, but not yet established yet.