# Design Document

By Barnabas Soon

My CS50 SQL Final project is a database for a (fictitious) single player card battler RPG video game called Battle Card Campus. Battle Card Campus is a single player card battling role playing game that takes place in a world where the card game Rift Mage is extremely popular. The game is inspired by the digital versions of collectible battle card games such as Magic The Gathering, Pokemon The Card Game and Yu-Gi-Oh! and also Japanese role playing games like Final Fantasy and Dragon Quest.

The player is a student attending Coral City Campus which is going to be holding a Rift Mage Card Tournament. Your goal is to defeat enough high ranking players to qualify for the Final Grand Campus Tournament, and become the #1 Rift Mage player on Campus! Just like in collectible card games like Pokemon The Card Game or  you will need to collect new cards to increase your To do so you're going to have to explore the campus, defeat other players to improve your ranking, collect cards and build a powerful deck.

## Video link

A short presentation on it can be reviewed here:
https://youtu.be/RD6125mPWF4?si=zqjsgckrqtwsstVk

## Scope

There are several core pillars of gameplay in the game:
**Combat:** Challenge others to a card battle to win battle points (BP), items and cards to climb in the rankings.
**Collecting:** Using your battle points to buy booster packs or single cards from sellers to add to your library of cards.
**Crafting:** Creating a better deck with your new cards so you can defeat opponents more easily.
**Quests:** There are main story quests that are required to progress the story and unlock new locations, cards and battle more powerful opponents. Completing minor quests will also get you more battle points, booster packs, items and cards which will make defeating enemies easier.

## Functional Requirements

The game will be played using a menu driven interface to explore the world, talk to other characters in the game and battle.

This is a single player game that is installed on the user's computer and there are no online multiplayer components that need to be considered.
The game supports multiple player profiles and saves.
The game will contain information about the game world such as dialogue, enemy decks and cards.

As the player completes main story quests the game will progress and new cards, locations and enemies will be unlocked.
The game will keep track player information such as the cards they own, battle points and decks constructed. Tables related to player progression or the players cards and items will need to be indexed.
The game will automatically save player information at certain points in the game such as when they finish battling or update their card decks. The player can also manually save their game.

## Battle Card Campus and Rift Mage

The section outlines relevant rules and information of Battle Card Campus and Rift Mage that will need to be stored in the database.

### Library and Deck Construction

Players must choose to add 20 to 30 cards to their deck at the start of the game.
There can be a maximum of 3 copies of a card in a deck. Therefore the player will never have more than 3 copies of a single card in their library.

### Cards

Players will play cards from their deck by paying the energy cost.
Cards can be a Creature card, Spell or Location card.
Cards can also have keywords such as 'Beast', 'Machine', 'Fire' etc. which might affect certain abilities or spells.
Creatures have health and attack and have special abilities.
Spells provide special effects or abilities.
Locations have health and special abilities but can not attack.

**Example cards:**

- **Name:** War Tiger
- **Set:** Wild Beasts
- **Battle Points cost:** 500
- **ID:** WLBT001
- **Type:** Creature
- **Keyword:** Beast
- **Energy cost:** 1
- **Health Points:** 40
- **Attack:** 20
- **Defense:** 0
- **Ability:** Can attack twice in a round.

- **Description:** A tiger trained for war.

The War Tiger is a card from the Wild Beasts card set that costs 500 Battle points to buy and has a serial number of WLBT001, Wild Beast set, card number 1.
War Tiger is a Creature card with the keyword beast. The card costs 1 to summon during battle.
It has 40 Health, an Attack of 20 and has a Defense of 0.
It has a special ability to attack twice during a round.
It has the text ‘A tiger trained for war.’ as part of the card

- **Name:** Fireball
- **Set:** Core
- **Battle Points Cost:** 500
- **ID:** CORE001
- **Type:** Spell
- **Keyword:** Fire
- **Energy cost:** 1
- **Ability:** Deals 50 damage to target Creature or Location
- **Text:** A giant spell that explodes and burns everything.

The Fireball card is a card from the Core card set that costs 500 Battle Points to buy and has a serial number of CORE005, Core set, card number 5.

A Fireball card is a Spell type card with a keyword of fire. It costs 1 to use in battle and summons a Fireball that deals 50 damage to a creature or Location.
It has the text: ‘A giant spell that explodes and burns everything.’

- **Name:** Wild Forest
- **Set:** Wild Beasts
- **Battle Points Cost:** 300
- **ID:** WLBT010
- **Type:** Location
- **Keyword:** None
- **Energy cost:** 0
- **Ability:** +20 Attack and +20 extra Health to all Beast Creatures.
- **Text**: Beasts can hide in the forest, giving them more attack and health.

The Wild Forest is a card from the Wild Beasts card set that costs 300 Battle points to buy and has a serial number of WLBT010, Wild Beast set, card number 10.

The Wild Forest card is a Location type card that grants extra attack and health to all Beast Creature type cards such as the War Tiger. The card costs 0 to summon during battle.

It has the text: ‘Beasts can hide in the forest, giving them more attack and health.’

## Characters

Characters can do 1 or more of the following:

- Talk to the player
- Battle the player
- Sell them cards or booster packs
- Offer them a story quest or minor quest

As the game's story or minor quests are completed, new characters will appear in their region. The same character may also change as the game progresses and offer different dialogue, update their deck with more powerful cards or sell different cards. In the database, they will be a new character with the same name but different id.

For example, the Jeff the Reckless Old Man at the start of Chapter 1 will start off with a deck of Beast type cards at the start of the game but in Chapter 2 as the game progresses he will add new Beast cards and Plant cards from his deck.

To represent this, Jeff the Reckless Old Man in Chapter 1 and 2 will have different entries in the character table. Jeff (Chapter 1, Version 1) will be unlocked at the start of the game and be replaced in the game with Jeff (Chapter 2, Version 2).

Game Text
The wil have lots of dialogue, game texts and information that will be need to be stored.

## Places

Places are divided into:
Region: Region of the Campus e.g. Engineering department, Science Department
Location: Locations will belong to a region and make up a region e.g. Science Department Offices, Lecture hall of the Engineering department.

# Entities

The tables are broadly divided into 3 categories.

Tables related to the cards and decks.

Tables related to the game world such as quests, places, characters and dialogue.

Tables related to the players, the progress they are making in the world and how many cards they have.

## Card tables

Tables related to cards and the decks.
Most of these columns will be NOT NULL unless otherwise stated.

### Cards

id: TEXT, PRIMARY KEY. The id of the card. As mentioned above, the card's serial number will be its id which is a mix of letters and numbers.
name: TEXT. Name of the card.
set_id: INTEGER. The card set this card will belong to. Foreign key constraint refers to the id on the Card Set table.
type: TEXT. It will be a ‘Creature’, ‘Spell’ or ‘Location’
rarity: TEXT. Rarity of the card. Will be listed as Common, Uncommon, Rare, Epic, Legendary.
battle_points_cost: INTEGER. How much Battle Points this card will cost to buy
energy_cost: INTEGER. How much energy this card will cost to play in the game.
health_points: INTEGER. How much life this will have if it’s a creature or location.
Attack: INTEGER. Attack points if it’s a creature or location. Can be NULL as Spells may not have Attack.
Defense: INTEGER. Defense points if it’s a creature of location. Can be NULL as Spells don’t have Defense.
ability01: TEXT. The first ability of the card. Can be null.
ability02: TEXT. Same as above. Can be null.
ability03: TEXT. Same as above. Can be null.
ability04: TEXT. Same as above. Can be null.
ability05: TEXT. Same as above. Can be null.

### Keywords

card_id:  TEXT. Foreign key constraint refers to the id on the Card table.
keyword: TEXT. Keywords on the card. There can be multiple keywords on a card.

### Card set

id: INTEGER, PRIMARY KEY. The id of the card set.
place_id: INTEGER. id of the region or location this can be bought. Foreign key constraint refers to the id on the Places table.
name: TEXT. Name of the set.
description: TEXT. Description of the set.
boosterpack_cost: INTEGER. Battle Points Cost to buy a booster pack from this set.

## Game World Tables

Tables related to the game world.

### Places

Contains information about Regions or Locations.
Most of these columns will be NOT NULL unless otherwise stated.

id: INTEGER, PRIMARY KEY. The id of the region or location.
type: TEXT. The type of location, either ‘Region’ or ‘Location’
region_id: INTEGER. If it’s a Location, it will belong to a Region. This is where the region ID will be if it’s location. Will be NULL if it’s a region. Foreign key constraint refers to the id on this table.
description: TEXT.  Description of the region or location.

### Game Texts

A place to store text in the game.

id: INTEGER, PRIMARY KEY. The id game text.
type: TEXT. Type of text, could be ‘dialogue’, ‘story’,’ flavour’, ‘battle’, ‘seller’,
part: INTEGER.  identifies which part of the story this belongs to. Can be NULL if this is not story specific.
chapter: INTEGER. identifies which story chapter in the story part his quest belongs to. Can be NULL if this is not chapter specific.
character_id: INTEGER. Identifies which character is saying dialogue. Can be NULL if it is not dialogue. Foreign key constraint refers to the id in Characters table.
quest_id: INTEGER.  Identifies which quest this is related to. Can be NULL if it is not quest text. Foreign key constraint refers to the id in Quests table.
previous_text_id: INTEGER. Links the previous text. Can be NULL if there is no previous text. Foreign key constraints refers to the id on this table.
next_text_id: INTEGER. Links the previous text. Can be NULL if there is no previous text. Foreign key constraints refers to the id on this table.
Text: TEXT. The text to be displayed.

### Quests

This table contains quest information. A quest/goal can have multiple sub quests/goals attached to it. Most of these columns will be not null unless otherwise stated.

id:  INTEGER, PRIMARY KEY. Quest id.
type: TEXT. The type of quest. Either ‘main’ story or ‘minor’ quest.
part: INTEGER.  Identifies which part of the story this belongs to
chapter: INTEGER.  Identifies which story chapter in the act this quest belongs to.
parent_id: INTEGER. This allows multiple quests to be linked together to a larger parent quest. Foreign key constraint refers to the id on this table.

previous_quest_id: Links quests together. This refers to the previous quest id.  Foreign key constraint refers to the id on this table.
next_quest_id: Links quests together. This refers to the next quest id. Foreign key constraint refers to the id on this table.
character_id: INTEGER. Identifies which character is giving the quest. Can be NULL if no specific character is giving it. Foreign key constraint refers to the id in the Character table.
place_id:  Identifies which place this quest is located. Foreign key constraint refers to the id in Places table.
text: TEXT. The description of the quest and of the reward offered.
completion_text: TEXT. The message you receive once you have completed the quest.

### Items

This table contains items that can be used to help the player battle or in the game world.

id: INTEGER. PRIMARY KEY. Item id.
name: TEXT. Name of the item
item_type: TEXT. Type of item. For example, ‘Battle’ for an item that can be used in battle. ‘Equipment’ for an item that can be equipped in the game.
cost: INTEGER. How much the item costs in battle points. Can be NULL or zero if the item can’t be bought.
item_slot: TEXT. Where the item fits in the equipment slot of a character. Can be NULL if this item can’t be equipped.
item_limit: INTEGER. Maximum number of this item that can be carried by the player.
description: Text. TEXT. Description of the item.

## Character Tables

Tables related to characters in the game.
Most of these columns will be NOT NULL unless otherwise stated.

### Characters

Information of the player and non-characters.
Most of these columns will be not null unless otherwise stated.

id:  INTEGER, PRIMARY KEY. id of character
name: TEXT. name of character
faction: TEXT. Refers to the faction this character belongs to.
enemy_id: INTEGER. Enemy id of this character if the character can be battled. Foreign key constraint refers to the id on the Enemies table.
seller_id: INTEGER. Seller id of this character if the character is a vendor and you can purchase items, cards or packs from the character. Foreign key constraint refers to the id on Sellers table.
version: INTEGER. Keeps track of which character version this is as the story progresses.
place_id: INTEGER Where the character appears. Foreign key constraint refers to the id on the Places table.
description: TEXT. Description of character.

### Enemies

Stores information about enemies.

character_id: INTEGER. Foreign key constraint refers to the id on the Characters table.
enemy_points: INTEGER. How many Battle Points you gain from defeating this enemy.
deck_name: TEXT. Name of the enemy's deck.
deck_description: TEXT. Description of the enemy's deck.
booster_pack_id: INTEGER. What card set the booster pack reward can be from. Can be null as some enemies might give you individual cards or items instead.  Foreign key constraint refers to the id on the card set table.
booster_pack_quantity: INTEGER. How many booster packs you get rewarded when defeating this enemy.

### Enemy Card List

Keeps track of cards in an enemy's deck and how many copies.

character_id:  INTEGER. Foreign key constraint refers to the id on the Characters table.
card_id: TEXT. Foreign key constraint refers to the id on the cards table.
card_copies: INTEGER. The number of card copies this deck has. Check constraint of 3 cards since that is the maximum number allowed in a deck.

## Enemy card rewards

Cards you can obtain by defeating an enemy.

character_id: INTEGER. Foreign key constraint refers to the id on the Characters table.
card_id: TEXT. Foreign key constraint refers to the id on the cards table.
card_copies: INTEGER. The number of cards that will be rewarded. Constraint of 3 cards in total since that is the maximum number allowed in a deck.

Enemy item rewards
Items you can obtain by defeating an enemy.

item_id: INTEGER. Foreign key constraint refers to the id on the items table.
item_copies: INTEGER. The number of items that will be rewarded.

Sellers card list
Cards you can obtain from sellers.

character_id: INTEGER. Foreign key constraint refers to the id on the Characters table.
card_id: TEXT. Foreign key constraint refers to the id on the cards table.
card_copies: INTEGER. The number of card copies this seller has. Check constraint of 3 cards since that is the maximum number allowed in a deck.

Seller item list
Items you can buy from sellers.

character_id: INTEGER. Foreign key constraint refers to the id on the Characters table.
item_id: INTEGER. Foreign key constraint refers to the id on the Items table.
items_copies: INTEGER. The number of items this seller has to sell.

## Player tables

Tables related to player information such as save files. Most will be NOT NULL.

### Players

Table containing the players information

id: INTEGER. id of the player. Maximum of 10 player profiles can be created in a game.
name: TEXT Name of the player
battles: INTEGER. Number of battles. Default is 0.
battles_won: INTEGER Number of battles won. Default is 0.
battles_lost: INTEGER Number of battles Lost. Default is 0.
game_time: INTEGER. How long the player has been playing the game.  Default is 0 seconds.

Player Save Files
Saves files for each player.

id: INTEGER. PRIMARY KEY. id of the save file.
player_id: INTEGER. id of the player. Foreign key constraint refers to the id on the Players table.
name: TEXT Name of the save file. Can be Null if no name is given.
save_file_date: INTEGER. Date and time of the save file.

### Player Card Library

This tracks which cards are in a player's card library.

player_id: INTEGER. Foreign key constraint refers to the id on the Players table.
save_id: INTEGER. Foreign key constraint refers to the id on the Player Save Files table.
card_id: TEXT. ID of the card
card_copies: how many copies of the card are owned.

Player Card Decks
Card decks saved by different players.

id: INTEGER. PRIMARY KEY. id of deck.
player_id: INTEGER. id of the player. Foreign key constraint refers to the id on the Players table.
save_id: INTEGER. id of the save file.
name: TEXT. Name of the deck.
date_created: TEXT. Date for when it was created. Stored in *YYYY-MM-DD HH:MM*
description: Text. Description. Can be NULL if no description entered.

### Player Card List

Card list for each deck.

deck_id: INTEGER. Foreign key constraint refers to the id on the Player Card Deck table.
card_id: TEXT. Foreign key constraint refers to the id on the Cards table.
card_copies: INTEGER. The number of card copies this deck has. Check constraint of 3 cards since that is the maximum number allowed in a deck.

### Player Owned Items

This is the table for whether a player has an item.
player_id: INTEGER. Id of the player. Foreign key constraint refers to the id on the Players table.
save_id: INTEGER. Save id of the player. Foreign key constraint refers to the Save Files id on the Players table.
Item_id: INTEGER. item id. Foreign key constraint refers to the id on the items table.
item_copies: INTEGER. Copies of item.

### Player Seller Cards

As a player buys cards, the seller's stock of cards will decrease. This keeps track of how many cards the seller has left.

player_id: INTEGER. Id of the player. Foreign key constraint refers to the id on the Players table.
save_id: INTEGER. Id of the player. Foreign key constraint refers to the id on the Player Save Files table.
character_id: INTEGER. Id of the player. Foreign key constraint refers to the id on the Characters table.
card_id:  TEXT. Id of the card. Foreign key constraint refers to the id on the Cards table.
card_copies: INTEGER. Number of cards left. Maximum should be 3.

### Player Seller Items

player_id: INTEGER. Id of the player. Foreign key constraint refers to the id on the Players table.
save_id: INTEGER. Id of the player. Foreign key constraint refers to the id on the Player Save Files table.
character_id: INTEGER. Id of the player. Foreign key constraint refers to the id on the Characters table.
item_id:  INTEGER. Id of the card. Foreign key constraint refers to the id on the Items table.
item_copies: INTEGER. Number of items left.

## Player Quests

This table tracks which player has completed what quests. Quests that are completed will show up here.

player_id: INTEGER. Id of the player. Foreign key constraint refers to the id on the Players table.
player_save_id: INTEGER Id of the player. Foreign key constraint refers to the id on the Player Save Files table.
quest_id:  Id of the Quest. Foreign key constraint refers to the id on the Quests table.

Player Places
Tracks what places have been unlocked.

player_id: INTEGER Id of the player. Foreign key constraint refers to the id on the Player Save Files table.
player_save_id: INTEGER Id of the player. Foreign key constraint refers to the id on the Player Save Files table.
place_id: INTEGER Id of the place. Foreign key constraint refers to the id on the place table.

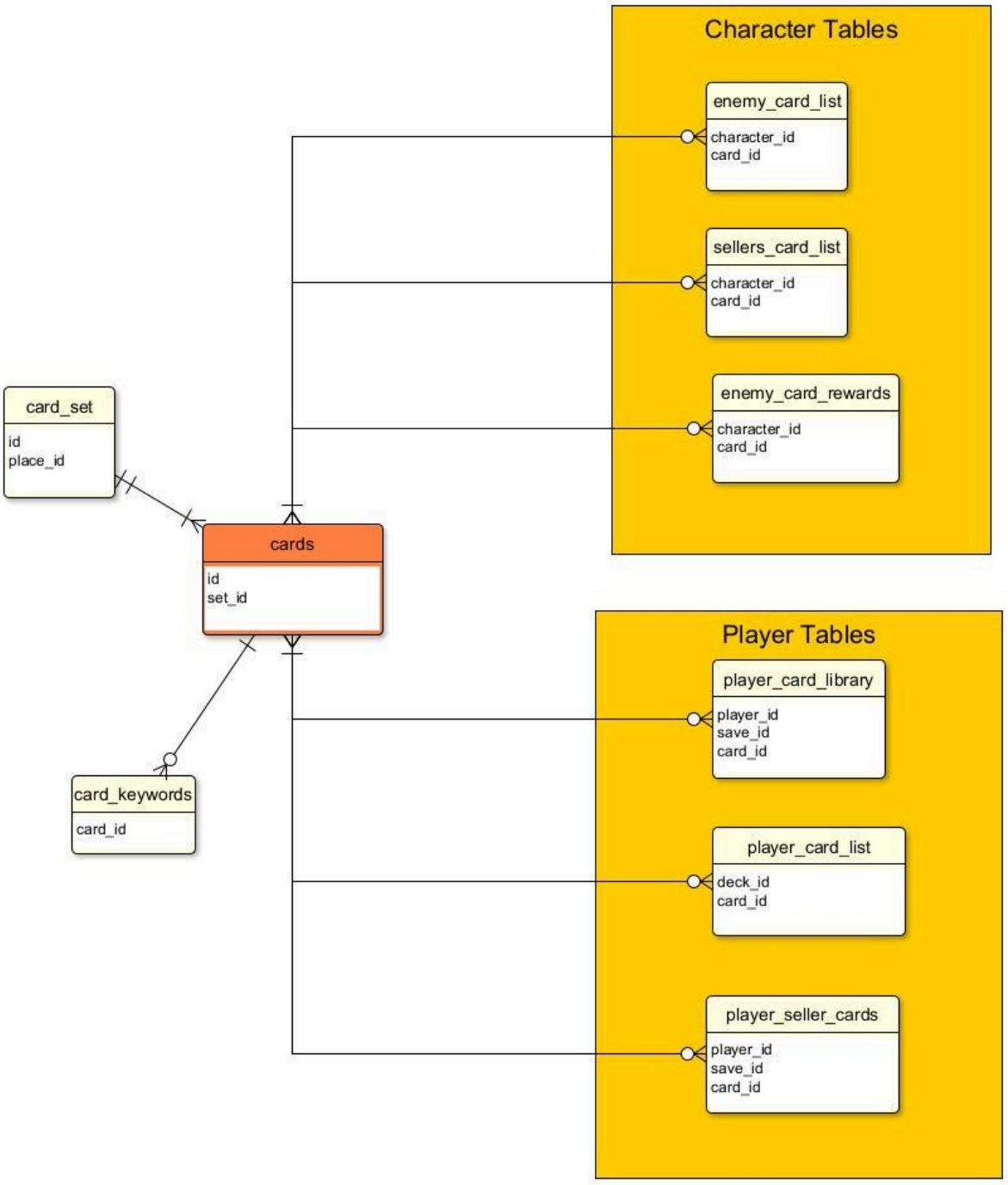
## Player Characters

player_id: INTEGER Id of the player. Foreign key constraint refers to the id on the Player Save Files table.
player_save_id: INTEGER Id of the player. Foreign key constraint refers to the id on the Player Save Files table.
character_id: INTEGER Id of the character. Foreign key constraint refers to the id on the place table.

# Relationships

These two tables highlights some of the most important tables and their relationship: Cards and Player tables

## Cards



Cards are the heart/core of the gameplay.

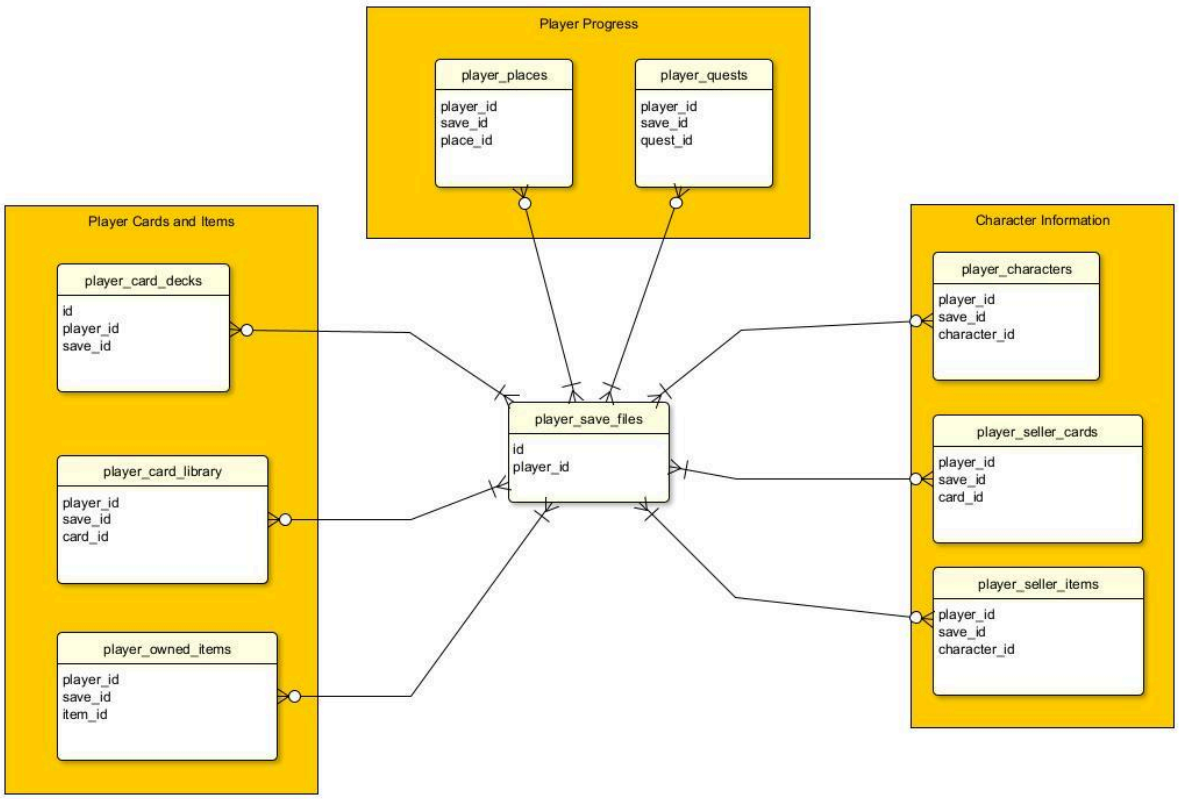
Cards all belong to a card set and can have 0 keywords or multiple keywords as part of the card.

Characters can be enemies or sellers or neither (a character that only talks to the player). Enemy characters will carry cards to use against the player which is shown in the Enemy Card List table and may give cards as a reward if they are beaten in combat.
Seller characters will sell cards to the player which is shown in the Sellers card list table.

Cards can also show up in the player tables. The player card library and player card list tables record the cards a player will get during the game and the decks of cards they have constructed respectively.

Finally, the player seller cards are there to record how many cards sellers in the player's game have left.

## Player Tables




Each player can have multiple player save files. Each save file will track the state of player progress in the world.

The player cards and items tables track what cards are in the current player's deck and library and the items they have.
The Player progress tracks the Quests and Places the player has unlocked.

The Character Information tables track characters the player has unlocked and the seller cards and items that have been bought.

# Optimisations

The game centres around cards
As a role playing video game that's centred around cards, battling and player progress, there's going to be a lot of accessing of the cards, characters and the player information to track the player progress in the world.
The following tables would benefit from indexing:

- Cards
- Game Texts
- Characters
- Player tables related to cards or quests.

# Limitations

The design is for a single player rpg and does not have tables for any multiplayer experience
This is designed primarily for a menu and text interface and may need to be expanded to include more information if this was a video game world which the player character can walk around it.