Table of Contents

Executive Summary
Entity-Relationship Diagram
Tables:
- Set
- Card
- Deck
- DeckList
- JudgeLevel
- GameType
- Tier
- Tournament
- Person
- Judge
- Player
- Match
- PlaysIn
- BannedPeople

Views
Reports
Stored Procedures
Triggers
Security
Implementation Notes
Known Problems
Future Enhancements
Executive Summary

The Magic the Gathering Organized Play database was created to help Wizards of the Coast, tournament organizers, and most importantly players organize officially sanctioned Magic the Gathering events. To this end, information on players, tournaments, judges and popular decks are all easily available. This allows players to view their matches, judges to rule on those matches, and tournament organizers to schedule their events and matches. This will allow Wizards of the Coast to more easily calculate player season points to determine invites to their high level tournaments around the world.
Entity-Relationship Diagram
Tables
Table: Set

This is a table stores the sets released for Magic the Gathering as well as the year they were released.

CREATE TABLE Set (  
id SERIAL,  
name text not null,  
year integer not null,  
primary key (id)  );

Dependencies: id -> name, year
Table: Card

This is a table stores the name and set id of cards.

CREATE TABLE Card (  
id      SERIAL,  
  name    text NOT NULL,  
  setID   INTEGER NOT NULL REFERENCES Set(id),  
  PRIMARY KEY (id)  
);

Dependencies: id -> name, setid
Table: Deck

In this table the name and set id of decks are stored.

CREATE TABLE Deck ( 
    id SERIAL, 
    name text, 
    PRIMARY KEY (id) 
);

Dependencies: id -> name
Table: DeckList

This table lists all the cards that make up a predefined deck. Instances refer to the number of times a given card is in a deck. Because the rules for Magic the Gathering limit a player to 4 copies of the same card per deck, instance must be a number between 1 and 4.

CREATE TABLE DeckList (  
id INTEGER REFERENCES Deck(id),  
cardID INTEGER NOT NULL REFERENCES Card(id),  
instances INTEGER NOT NULL DEFAULT 1 CHECK (instances >= 1 AND instances <= 4),  
PRIMARY KEY (id, cardID)  );

Dependencies: id -> cardid, instances
Table: JudgeLevel

This table contains information on the judge levels for sanctioned Magic the Gathering events. The level, name of the level, and the how long the licence is valid before needing to be renewed are stored here.

CREATE TABLE JudgeLevel (
    level INTEGER NOT NULL,
    name text NOT NULL,
    yearsValid INTEGER NOT NULL,
    PRIMARY KEY (level)
);

Dependencies: level -> name, yearsValid
Table: GameType

The GameType table stores the current sanctioned game types for competitive Magic the Gathering.

CREATE TABLE GameType (  
  id SERIAL,  
  name text NOT NULL,  
  PRIMARY KEY (id)  
);

Dependencies: id -> name
The Tier table stores the name and weight of Magic the Gathering tournament tiers. The weighting is important for calculating players’ league points based on the tournaments they participated in.

CREATE TABLE Tier (  
id SERIAL,  
name text NOT NULL,  
weight INTEGER NOT NULL,  
PRIMARY KEY (id)  );

Dependencies: id -> name, weight
## Table: Tournament

This table stores the information on tournaments. Their names, locations, dates, and tier are all stored here.

```sql
CREATE TABLE Tournament (  
id SERIAL,  
name text NOT NULL,  
location text NOT NULL,  
date date NOT NULL DEFAULT current_date,  
tierID integer NOT NULL REFERENCES Tier(id),  
PRIMARY KEY (id)
);
```

Dependencies: id -> name, location, date, tierid

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>location</th>
<th>date</th>
<th>tierid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Super amazing tournament of champions</td>
<td>the moon</td>
<td>2017-12-01</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>FNM</td>
<td>Poughkeepsie, NY</td>
<td>2017-12-01</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Grand Prix Atlanta</td>
<td>Atlanta, GA</td>
<td>2017-12-01</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Grand Prix Warsaw</td>
<td>Warsaw, Germany</td>
<td>2017-12-01</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Iconic masters Prerelease</td>
<td>New York, NY</td>
<td>2017-12-01</td>
<td>4</td>
</tr>
</tbody>
</table>
Table: Person

This table stores basic information on a person. This does not guarantee they are a player, judge, or banned person.

CREATE TABLE Person (  
id SERIAL,  
fname text NOT NULL,  
lname text NOT NULL,  
bday INTEGER NOT NULL,  
bmonth INTEGER NOT NULL,  
byear INTEGER NOT NULL,  
PRIMARY KEY (id)
);

Dependencies: id -> fname, lname, bday, bmonth, byear
The judge table has information on people who are certified judges. Their judge level and year they were certified are stored here.

CREATE TABLE Judge (  personID INTEGER NOT NULL REFERENCES Person(id),  level INTEGER NOT NULL REFERENCES JudgeLevel(level),  certYear INTEGER NOT NULL,  PRIMARY KEY (personID) );

Dependencies: personid -> level, certYear
Table: Player

The Player table has information on people who are registered players.

CREATE TABLE Player (  
    personID INTEGER NOT NULL REFERENCES Person(id),  
    favCard INTEGER NOT NULL REFERENCES Card(id),  
    PRIMARY KEY (personID)  
);

Dependencies: personid -> favCard
Table: Match

The Match table holds information on individual matches such as the tournament they were played in, the game type, and the judge who presided over the match.

CREATE TABLE Match (    id SERIAL,    tournamentID INTEGER NOT NULL REFERENCES Tournament(id),    gameType INTEGER NOT NULL REFERENCES GameType(id),    judgeID INTEGER NOT NULL REFERENCES Judge(personID),    PRIMARY KEY (id) );

Dependencies: id -> tournamentid, gameType, judgeid
Table: PlaysIn

This table links players to matches. Each player in a match has their deckid saved as well as their side. This is important for non-traditional game modes like 2 Headed Giant where multiple people play on the same side. It also stores the result of the match. The result can be null denoting a game that is scheduled but has not occurred, or that is in the process of being played.

CREATE TABLE PlaysIn (  
    matchID INTEGER NOT NULL REFERENCES Match(id),  
    playerID INTEGER NOT NULL REFERENCES Player(personID),  
    deckID INTEGER NOT NULL REFERENCES Deck(id),  
    side integer NOT NULL,  
    result CHAR(1) CHECK (result = 'W' or result = 'L'),  
    PRIMARY KEY (matchID, playerID)  
);
Table: BannedPeople

This table stores people who have been banned from Wizards of the Coast sanctioned events. The reason for the ban, the start date, and the length are found here.

CREATE TABLE BannedPeople ( 
  personID INTEGER NOT NULL REFERENCES Person(id),
  reason text,
  startDate date NOT NULL DEFAULT current_date,
  length_months INTEGER NOT NULL,
  PRIMARY KEY (personID)
);

Dependencies: personid -> reason, startDate, length_months
Views
View: fullMatchView

This view shows the full information on a match, the players, their sides, the judge, result, and game type.

CREATE VIEW fullMatchView AS
SELECT match.id as match_id, PlaysIn.playerID as player_id,
    p1.fname as player_name, PlaysIn.side,
    judge.personID as judge_id,
    p2.fname as judge_name, result, gameType.name
FROM Match INNER JOIN PlaysIn ON Match.id = PlaysIn.matchID
    INNER JOIN Player ON PlaysIn.playerID = Player.personID
    INNER JOIN Judge ON Match.judgeID = Judge.personID
    INNER JOIN person as p1 ON Player.personID = p1.id
    INNER JOIN person as p2 ON Judge.personID = p2.id
    INNER JOIN gametype ON Match.gameType = GameType.id;

<table>
<thead>
<tr>
<th>match_id</th>
<th>player_id</th>
<th>player_name</th>
<th>side</th>
<th>judge_id</th>
<th>judge_name</th>
<th>result</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Anders</td>
<td>1</td>
<td>2</td>
<td>Richard</td>
<td>W</td>
<td>Standard</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Bob</td>
<td>2</td>
<td>2</td>
<td>Richard</td>
<td>L</td>
<td>Standard</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Anders</td>
<td>1</td>
<td>2</td>
<td>Richard</td>
<td>&lt;null&gt;</td>
<td>Standard</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Some</td>
<td>2</td>
<td>2</td>
<td>Richard</td>
<td>&lt;null&gt;</td>
<td>Standard</td>
</tr>
</tbody>
</table>
View: bannedPeopleinfo

This table stores people who have been banned from Wizards of the Coast sanctioned events. The reason for the ban, the start date, and the length are found here.

CREATE VIEW bannedPeopleInfo AS
SELECT personID, fname, lname, startDate, length_months
FROM BannedPeople INNER JOIN person ON
    BannedPeople.personID = Person.id;

<table>
<thead>
<tr>
<th>personid</th>
<th>fname</th>
<th>lname</th>
<th>startDate</th>
<th>length_months</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bad</td>
<td>Guy</td>
<td>2000-01-01</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Mark</td>
<td>Watson</td>
<td>2017-12-01</td>
<td>12</td>
</tr>
</tbody>
</table>
Reports
Report: DeckList
Card Counts

This report shows the total counts of a card across all decks and sorts by highest amount first.

SELECT cardID, sum(instances) as rate
FROM DeckList
GROUP BY cardID
ORDER BY rate DESC;
Report: Currently Banned People

This report shows the full details on people whose bans are still in effect. That is the duration of the ban is not complete.

```
SELECT id, fname, lname, reason, startDate, legth_months, age(current_date, startDate)
FROM bannedpeople INNER JOIN Person ON BannedPeople.personID = Person.id
WHERE extract(YEAR FROM age(current_date, startDate)) * 12 + extract(MONTH FROM age(current_date, startDate)) < legth_months;
```
Stored Procedures
Stored Procedure:
get_matches_by_tournament

This stored procedure will get all matchid’s for a given tournament.

CREATE OR REPLACE FUNCTION get_matches_by_tournament(text) RETURNS TABLE (matchID INTEGER) AS $$
DECLARE
    tournamentName ALIAS FOR $1;
BEGIN
    RETURN QUERY
        SELECT DISTINCT Match.id
        FROM Match INNER JOIN tournament ON Match.tournamentID = Tournament.id
        WHERE tournament.name = tournamentName;
END;
$$LANGUAGE plpgsql;
Stored Procedure: 
get_points_for_player

This stored procedure will calculate the points a player has earned from all the tournaments they have participated using the tournament weight.

CREATE OR REPLACE FUNCTION get_points_for_player(text, text) RETURNS TABLE (points BIGINT) AS $$
DECLARE
    firstName ALIAS FOR $1;
    lastName ALIAS FOR $2;
BEGIN
    RETURN QUERY
        SELECT coalesce(sum(tier.weight), 0)
        FROM PlaysIn INNER JOIN Person ON playerID = id
            INNER JOIN match ON PlaysIn.matchID = Match.id
            INNER JOIN tournament ON Match.tournamentID = Tournament.id
            INNER JOIN tier ON Tournament.tierID = Tier.id
        WHERE fname = firstName
        AND lname = lastName
        AND result = 'W';
END;
$$ LANGUAGE plpgsql;
Triggers
This trigger makes sure someone who is a judge is not scheduled to play in a match they also officiate.

```sql
CREATE OR REPLACE FUNCTION check_match_judge()
RETURNS TRIGGER AS
$$
BEGIN
  IF exists(SELECT * FROM match WHERE Match.id = NEW.matchID AND match.judgeID = NEW.playerID) THEN
    RAISE NOTICE 'Cannot judge and play in the same match';
    RETURN NULL;
  END IF;
  RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER check_match_judge
BEFORE INSERT OR UPDATE ON PlaysIn
FOR EACH ROW
EXECUTE PROCEDURE check_match_judge();
```

```sql
sql> INSERT INTO playsin (matchid, playerid, deckid, side, result) VALUES (2, 2, 1, 1, NULL)
[2017-12-02 22:14:07] [000000] Cannot judge and play in the same match
[2017-12-02 22:14:07] completed in 7ms
```
Trigger: check_deck_size

This trigger makes you cannot put in more than 40 cards into a decklist as per Magic the Gathering's standard tournament rules.

CREATE OR REPLACE FUNCTION check_deck_size()
RETURNS TRIGGER AS
$$
DECLARE
  numCards INTEGER := 0;
BEGIN
  SELECT INTO numCards sum(instances) FROM DeckList WHERE NEW.id = DeckList.id;

  IF numCards + NEW.instances > 60 THEN
    RAISE NOTICE 'There are too many cards in that deck.';
    RETURN NULL;
  END IF;
  RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER check_deck_size
BEFORE INSERT OR UPDATE ON DeckList
FOR EACH ROW
EXECUTE PROCEDURE check_deck_size();
Security
Rolls

The current rolls are admin, tournamentAdmin, judge, and player. Start with removing everyone's abilities then add back in admin to have full power.

```sql
CREATE ROLE admin;
CREATE ROLE tournamentAdmin;
CREATE ROLE judge;
CREATE ROLE player;

--
-- Remove access from all rolls before restoring it.
--
REVOKE ALL ON ALL TABLES IN SCHEMA public from admin;
REVOKE ALL ON ALL TABLES IN SCHEMA public from tournamentAdmin;
REVOKE ALL ON ALL TABLES IN SCHEMA public from judge;
REVOKE ALL ON ALL TABLES IN SCHEMA public from player;

--
-- Give admin back power.
--
GRANT ALL ON ALL TABLES IN SCHEMA public to admin;
```
Rolls: Player

The player roll has control over deck and decklist allowing them to edit their decks. They can also view all the cards from all the sets as well as see the matches they are scheduled to play in.

GRANT SELECT, INSERT, UPDATE, DELETE ON Deck to player;
GRANT SELECT, INSERT, UPDATE, DELETE ON Decklist to player;
GRANT UPDATE ON Person to player;
GRANT UPDATE ON Player to player;
GRANT SELECT ON Card TO player;
GRANT SELECT ON Set TO player;
GRANT SELECT ON Match TO player;
GRANT SELECT ON PlaysIn TO player;
GRANT SELECT ON Tournament TO player;
GRANT SELECT ON GameType TO player;
GRANT SELECT ON Tier TO player;
Rolls: Judge

The judge roll has much the same power as the player, but they are also allowed to ban people change information about matches, and other judges.

GRANT SELECT, INSERT, UPDATE ON BannedPeople to judge;
GRANT SELECT, UPDATE ON Match to judge;
GRANT SELECT, UPDATE ON PlaysIn to judge;
GRANT SELECT, UPDATE ON Judge to judge;
GRANT SELECT ON Deck TO judge;
GRANT SELECT ON DeckList TO judge;
GRANT SELECT ON Card TO judge;
GRANT SELECT ON Set TO judge;
GRANT SELECT ON Match TO judge;
GRANT SELECT ON PlaysIn TO judge;
GRANT SELECT ON Tournament TO judge;
GRANT SELECT ON GameType TO judge;
GRANT SELECT ON Tier TO judge;
Rolls: TournamentAdmin

The routnamentAdmin roll is an extension of the judge. They are given all the same power as a judge including banning people, but are also allowed to edit information on the tournaments themselves. They are not however allowed to change anything about tournament tiers, or gametypes.

GRANT SELECT, INSERT, UPDATE ON BannedPeople to tournamentAdmin;
GRANT SELECT, INSERT, UPDATE ON Match to tournamentAdmin;
GRANT SELECT, INSERT, UPDATE ON PlaysIn to tournamentAdmin;
GRANT SELECT, INSERT, UPDATE ON Tournament to tournamentAdmin;
GRANT SELECT, UPDATE ON Judge to tournamentAdmin;
GRANT SELECT ON Deck TO tournamentAdmin;
GRANT SELECT ON DeckList TO tournamentAdmin;
GRANT SELECT ON Card TO tournamentAdmin;
GRANT SELECT ON Set TO tournamentAdmin;
GRANT SELECT ON Match TO tournamentAdmin;
GRANT SELECT ON PlaysIn TO tournamentAdmin;
GRANT SELECT ON Tournament TO tournamentAdmin;
GRANT SELECT ON GameType TO tournamentAdmin;
GRANT SELECT ON Tier TO tournamentAdmin;
Implementation Notes and Known Problems
Implementation and Known Problems

- Currently player points are only awarded for a win. Something to look at is if base points should be given just for attending events, or if loses should award a small percentage of points a win would have been worth. Those are both things that if ruled on would require changing in the get_points_for_player function.

- Currently there is no way to ban cards from decks. While it is very rare for a card to be banned in standard, it's very common for older game types with less rules on allowed sets.

- Currently there is nothing stopping a banned player from being a judge or a player. For now this means judges and tournament organizers need to check the currently banned people report to make sure they are not scheduling banned people.
Future Enhancements
Future Enhancements

- Include more information on cards. Currently we only store card names and the sets they are from. Going forward it could be beneficial to store information such as rarity, mana-cost, mana-color, attack, defence, and special effects.

- There is currently no way of informing judges their license is about to or has expired. This is another improvement that could be made.

- Storing things like player points or win/loses might also be beneficial. Instead of calculating them every time they are needed, having them update at the end of tournaments would be much better.

- Being able to separate promotional version of cards from various events or products would also be useful. This would give insight into the kinds of rewards our players like and want more of.

- The current definition for the standard game type is the 3 most recent blocks. In the past a block has been made of 3 sets, but recently Wizards of the Coast has released 2 set blocks. This makes it hard to track what cards are standard legal. A way to fix this issue would be to have an additional table listing the blocks and the sets that belong to them.