Jonathan Wiseman, Ian Bowler, Matthew Grohotolski, Ryan Ehmann

February 13, 2020

CS 341

Database Design

1. **Description**

LFJ is a Discord bot serving Elizabethtown College’s E-Sports team. It satisfies five goals given by the team:

1. Handling event signup
2. Balancing scrimmage teams
3. Posting reminders for registered events
4. Allowing users to self-assign to groups
5. Relevant statistics collection and querying

The system architecture consists of three primary components:

1. The Discord environment in which users can exchange messages, react to messages, and summon the bot via text and message reactions
2. The bot’s front end that receives events via the Discord API. Events include messages and reactions. Events are processed and handled by the bot’s backend scripts
3. The bot’s back end, consisting of Python scripts to handle events and access the backend database.

This document deals with the design of the bot’s back end databases. The database must handle goals number one, three, four, and five.

1. **Business Rules**

LFJ is driven primarily by user interaction with scheduled events and the games that they play.

* A user registers for zero or more events; an event consists of zero or more users.
  + This n:n relationship necessitates a bridge entity: registration
* A user plays zero or more games; a game is played by zero or more users
  + This n:n relationship necessitates a bridge entity: membership
* An event is about exactly one game; each game may be played in zero or more events
* Each registration produces a performance; each performance matches exactly one registration

1. **Entity Summary**

* user:
  + user\_id: the user’s unique integer identifier
  + display\_name: the user’s Discord display name
  + e-mail: an (optional) way to contact the user
* event:
  + event\_id: the event’s unique integer identifier
  + game\_id: the ID of the game being played in this event
  + date: the date of the event
* game:
  + game\_id: the game’s unique integer identifier
  + description: the game’s title (ex. League of Legends)
* membership:
  + user\_id: the ID of the user that plays the game
  + game\_id: the ID of the game that the user plays
  + skill\_level: a measure of the user’s skill at the specified game (used for matchmaking)
* registration:
  + user\_id: the ID of the user making the registration
  + event\_id: the ID of the event being registered for
  + date: date of the event being registered for
* performance:
  + user\_id: the ID of the user who competed
  + event\_id: the ID of the event that was competed in
  + kills: measure of positive performance
  + deaths: measure of negative performance
  + win: did the player win this event?
  + length: length of the match
  + win\_score: score of the winning team
  + lose\_score: score of the losing team

1. **Entity-Relation Diagram**A screenshot of a cell phone

   Description automatically generated