Dustin Alin and Nicholas Gordon 10/14/13

CPS 316 Final Project - Milestone 1 Prof. Yang

**Platform Description**

We will be creating an iOS application. Consequently, the application will be written with Objective-C. We have decided to use PostgreSQL as our database management system. The production dataset we are using is obtained through XML documents, so we use the Java implementation of DOM to process the XML before transforming the data to SQL. The process of doing this is described in our README. To communicate with the PostgreSQL server, our application will use an open-source framework known as PGClientKit. This framework utilizes the PostgreSQL library lipbq.

**Constraints & Assumptions**

1. Each team has exactly one school, represented in the database as an alias. A team’s name is not unique. A team is identified by its school.
2. A player can only play for at most one team at any given time. He may not be playing for any team i.e. a college recruit who hasn’t signed a letter of intent, red-shirt freshman, graduate. A player may have played for multiple teams i.e. a transfer.
3. A player has exactly one position. While in real life a player may have multiple positions, we assume he has exactly one, representing his natural position, regardless of what position he may be used as during a game.
4. Two players may have the same first and last name.
5. A venue does not necessarily belong to any school, as it could be a neutral site, i.e. MSG. A venue may belong to more than one team.
6. A game is played at exactly one venue.
7. A team plays in exactly one conference.
8. Every player, venue, conference, team, and game has a unique id.

1. A game must have two Division 1 teams. A game cannot have one Division 1 team and a team from another division.
2. A player’s statistics for a given game are uniquely identified by the player’s id and the game’s id.

**Relational Schema**

Venue(id, name, address, city, state, country, zipcode, capacity)

Conference(id, name, alias)

Team(id, name, alias, venue\_id, conference\_id)

Game(id, home\_team\_id, away\_team\_id, scheduled\_date, scheduled\_time)

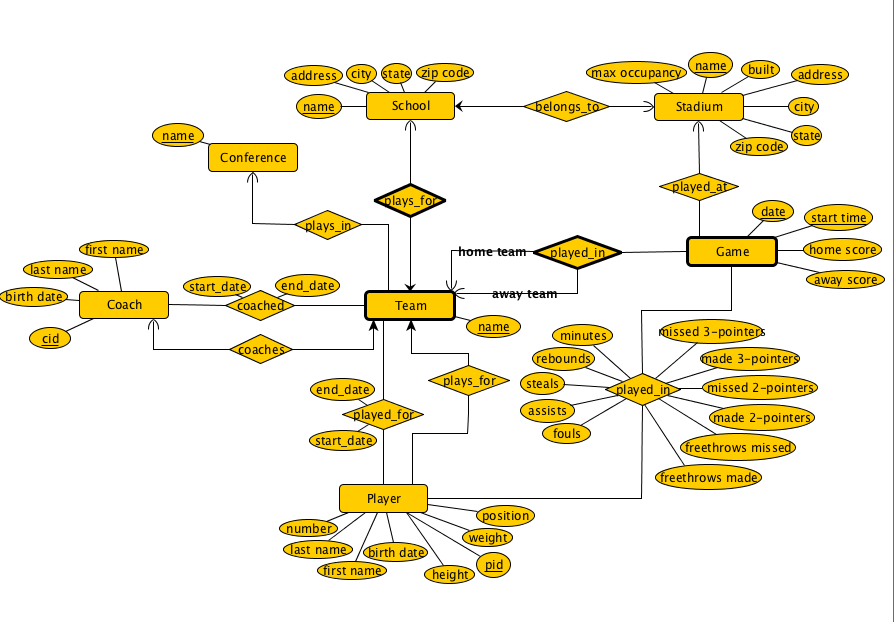
Player(id, first\_name, last\_name, team\_id, height, weight, jersey\_number, year, position, birthplace)

GameStats(game\_id, player\_id, minutes, three\_point\_attempts, three\_point\_makes, two\_point\_attempts, two\_point\_makes, field\_goal\_attempts, field\_goal\_makes, free\_throw\_attempts, free\_throw\_makes, offensive\_rebounds, defensive\_rebounds, assists, turnovers, steals, blocks, personal\_fouls, technical\_fouls, flagrant\_fouls, points)

Score(game\_id, home\_score, away\_score)

**Database Changes**

After reviewing the production dataset further, we modified our relations to have more attributes based on data that we saw in the dataset we did not include previously. For example, we added more attributes to GameStats after noticing that the dataset included breakdowns for the rebounds (offensive and defensive) and fouls (personal and technical). We also removed some relations because of a lack of information provided. The dataset did not contain what teams the players and coaches were on in the past. Consequently, we removed the relations PlayedFor and Coached, which contained this information. We also removed the relation Coach. Additionally, we decided to create an id attribute for Venue, Conference, Team, Game, and Player. Finally, we added a Score relation. While it would have been better to put the score information in the Game relation, we ran into efficiency issues when processing the XML documents, so we decided that this was a necessary tradeoff.

**E/R Diagram**