Abhilash Vallamkonda
126004453

Project 1 Report

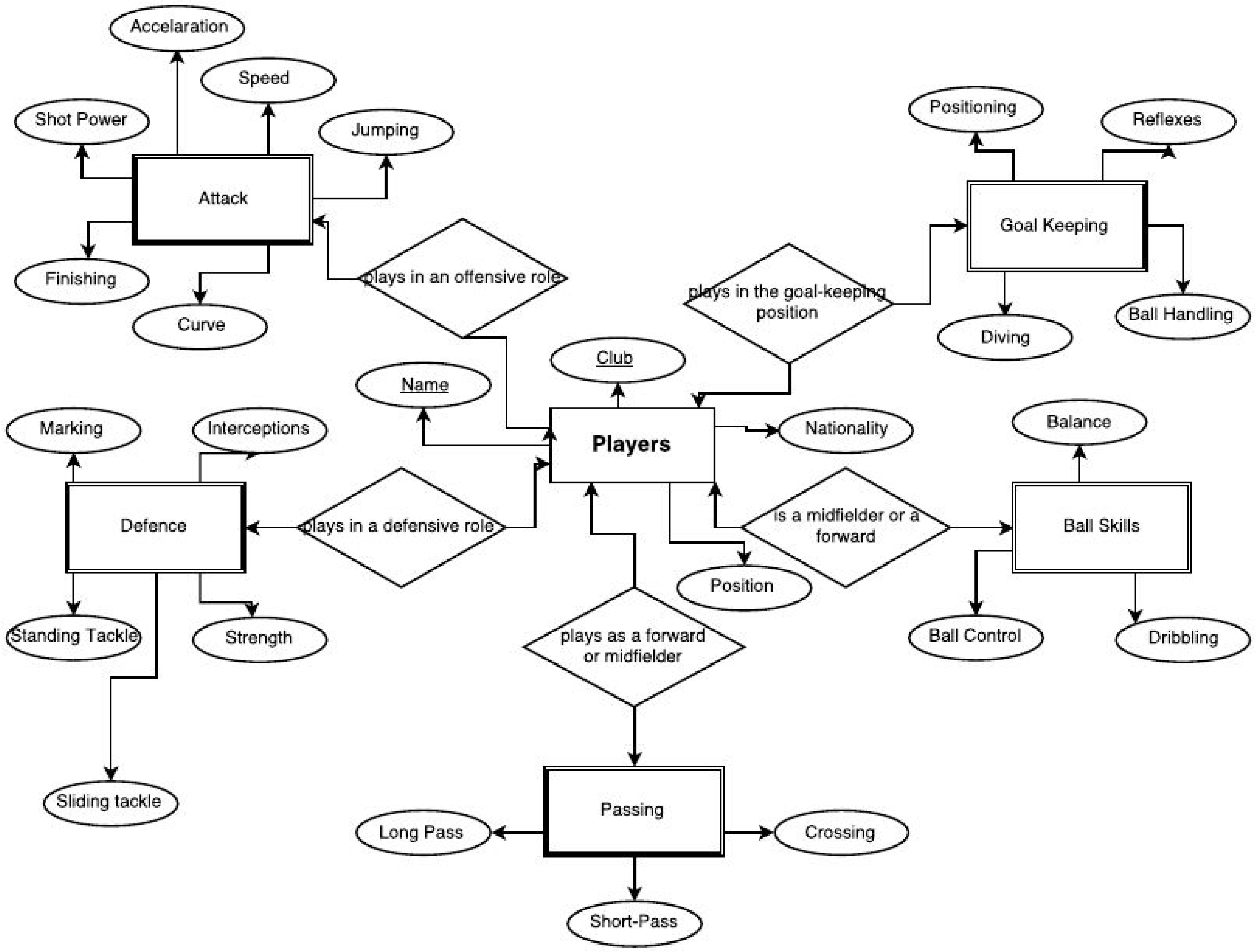
Project Description:

I find it strange that there is no website that combines the different soccer leagues from around the world. The best site for fantasy soccer right now is the Fantasy Premier League which only focuses on the EPL(English Premier League). The La Liga, Serie A and Bundesliga are equally great as the EPL and there really is no reason why there can't be a fantasy league which includes these as well. A lot of these matches overlap and being able to choose players from around the world definitely makes the fantasy league more interesting. My project is to design a Fantasy Soccer League website which includes a database of 15000+ soccer players from around the world.

Data Collection:

Soccer is the most popular game on this planet and there are a lot of soccer players. This makes data collection a real challenge. Generating fake data is not a great idea since being able to choose your favorite players is what makes fantasy leagues so much fun! Luckily I was able to find a dataset which basically contains all the data from the Fifa 17 videogame. This is great because the FIFA 17 game contains the most accurate representation of the the real world soccer teams that I know of. Still, there is a lot of scope for improvement and my website provides an opportunity for the users to contribute to the database in addition to playing in the fantasy soccer league.
There is still one issue though, the dataset contains a lot of data about every player and not all of it is useful in helping the user decide whether or not to include someone on their fantasy league team. For eg, The dataset provides a "Sliding Tackle" attribute which simply indicates that Cristiano Ronaldo has a really bad sliding tackle. Unless the user is a soccer fan, it is hard to guess that this value actually doesn't mean anything since he is a striker whose role is to score as many goals as possible. So, the dataset must be pre-processed to remove all the unneeded and misleading data. The final structure is described in the ER-Diagram that follows.

E-R Diagram:



"Players" is the Entity set of all soccer players in the database. Every soccer player has a "Position" and this decides the attributes for the player. For eg, only a forward would have an entry in the "Attack" Entity set.

- All relationships are one-one and this is indicated by the bidirectional arrows (↔) connecting the entity sets.
- All Entity Sets other than "Players" i.e "Attack", "Defence", "Passing", "Goal Keeping" and "Ball Control" are weak entity sets.
- The "Players" Entity set has (Name, Club) of the player as the primary key.
- All other Entity Sets are weak and hence have the same primary key.

Entity Sets to Relations:

Converting the E-R diagram into a relations is fairly straightforward.

- The “Players” Entity Set and its attributes form the “Players” relation.
- For each of the remaining Entity set, a relation is created using all the attributes of the entity set + the key attributes of “Players”.

Table Normalization:

The only non-trivial FD (functional dependency) here is that the key consisting of (Name, Club) determines all other attributes for each of the relations.

(Name, Club) → All other attributes

It is obvious that the left side of this FD contains the key and this implies that the table structure is already in BCNF and no changes are needed.

User Interface:

The goal of the fantasy league is to choose a team of 11 players and depending on their performance in the actual soccer game, points are assigned. For eg, if a player on your team scores a goal, you get 5 points, 3 points for an assist and so on.
The chosen team must consist of 2 forwards, 4 midfielders, 4 defenders and 1 goalkeeper. The chosen players can belong to any of the clubs in any of the soccer leagues around the world. Points are assigned based on the team’s performance.
My website provides an interface to search for players from the database and add them to your team. The search query for the database uses a combination of mysql+php. The result of the query is displayed as a table on the webpage via the POST protocol. The option to add the player to the user’s team has been implemented using Javascript. The users must search the database for players they want to add to their team using the search query, then click on “Add to Team” which takes all the players in the result of the query to the team.
As described earlier, my website allows users access to the database and provides an opportunity for everybody to contribute. Users are allowed to “Insert”, “Update” or “Delete” from the database but only one entry can be modified per request. This is to discourage spammers and also guard against users accidentally corrupting the database. All operations are carried out using php to connect to the mysql database and all of the code is embedded in the html web page.

Implementation:

A python script is used to to remove the unneeded data from the initial dataset. This splits the initial dataset into 6 tables but note that all the unneeded data, for eg : the goal keeping data for forwards etc, is still in the tables.
The script produces 6 csv files, one for every relation. The csv files are then uploaded to the database server using mysql. This is where all the unneeded data is removed using SQL DELETE queries.

There is one important property which must be satisfied by the database: The Foreign key constraint for all tables other than general_info. This is essential to ensure that all the attributes correspond to some player present in the database. This ensures that the integrity of the database is maintained even with multiple “Insert” and “Delete” requests from the users. Please refer to the appendix for a list of the SQL commands used.

Discussion:

This project was an amazing learning experience for me. This was my first time using html, css, php, mysql and javascript. It seemed like a lot of work in the beginning but things got easier as I started to get the hang of it.
In general, this project gave me an idea of the challenges involved in creating a complete system and how things can go wrong when the individual components come together.
The first issue I encountered was in the character set of the database, it took me a while before I understood what the problem was after which I was able to set the encoding of the database to utf-8.
The other issue I had was that mariaDB does not allow assertions and reports a syntax error when it is used. This was baffling initially because there really wasn’t anything wrong with the syntax. Then, I learnt that none of the commercial databases implement assertions but figuring this out took a lot of time.
The most challenging aspect in designing a website was the lack of a debugging framework which I am used to while programming in languages like C++ or Python. There were multiple instances where my seemingly innocent change would result in a HTTP 500 internal server error offering no clue whatsoever about what the problem actually is. It took me sometime to get used to this and I settled into a process or making incremental changes and confirming that there are no bugs before proceeding at every stage of the design.

Appendix:

Only the SQL commands which I entered through the command line are included in this report, all other source files provided in a zipped file.

Commands:

```
mysql -h database2.cs.tamu.edu -u vrabhilash -p vrabhilash-project1

mysql -h database2.cs.tamu.edu -u vrabhilash -p vrabhilash-project1 -e "LOAD DATA LOCAL
INFILE 'table_name.csv' INTO TABLE table_name columns terminated by ',' lines terminated by
'\r\n' ignore 1 lines; SHOW WARNINGS"
```

Creating the tables

```
create table general_info(Name VARCHAR(50), Nationality VARCHAR(20), Club
VARCHAR(30), Position VARCHAR(15) NOT NULL, PRIMARY KEY(Name, Club))

create table ball_skills_all(Name VARCHAR(50), Club VARCHAR(30), Ball_Control TINYINT,
Dribbling TINYINT, Balance TINYINT,
   primary key(Name, Club),
   foreign key(Name, Club) references general_info(Name, Club));

create table ball_skills(Name VARCHAR(50), Club VARCHAR(30), Ball_Control TINYINT,
Dribbling TINYINT, Balance TINYINT,
   primary key(Name, Club),
   foreign key(Name, Club) references general_info(Name, Club));

INSERT INTO ball_skills
   SELECT Name, Club, Ball_Control, Dribbling, Balance
   FROM general_info NATURAL JOIN ball_skills_all
   WHERE Position LIKE 'Forward' or Position LIKE 'Midfielder';

create table passing_all(Name VARCHAR(50), Club VARCHAR(30), Short_Pass TINYINT,
Crossing TINYINT, Long_pass TINYINT,
   primary key(Name, Club),
   foreign key(Name, Club) references general_info(Name, Club));

create table passing(Name VARCHAR(50), Club VARCHAR(30), Short_Pass TINYINT,
Crossing TINYINT, Long_Pass TINYINT,
   primary key(Name, Club),
   foreign key(Name, Club) references general_info(Name, Club));

INSERT INTO passing
   SELECT Name, Club, Short_Pass, Crossing, Long_Pass
   FROM general_info NATURAL JOIN passing_all
   WHERE Position LIKE 'Forward' or Position LIKE 'Midfielder';
```

* The result of this query must be an empty set. This means that there are no forwards or midfielders who lack passing attributes

```
SELECT Name, Club
FROM general_info
WHERE (Position LIKE 'Midfielder' OR Position LIKE 'Forward') AND
   NOT EXISTS
      (SELECT Name, Club
      FROM passing
      WHERE general_info.Name LIKE Name AND
```

```
        general_info.Club LIKE Club
        );

create table attack_all(Name VARCHAR(50), Club VARCHAR(30),
    Jumping TINYINT, Heading TINYINT, Shot_Power TINYINT, Finishing TINYINT,
    Long_Shots TINYINT, Curve TINYINT, Acceleration TINYINT, Speed TINYINT,
    primary key(Name, Club),
    foreign key(Name, Club) references general_info(Name, Club));

create table attack(Name VARCHAR(50), Club VARCHAR(30),
    Jumping TINYINT, Heading TINYINT, Shot_Power TINYINT, Finishing TINYINT,
    Long_Shots TINYINT, Curve TINYINT, Acceleration TINYINT, Speed TINYINT,
    primary key(Name, Club),
    foreign key(Name, Club) references general_info(Name, Club));

INSERT INTO attack
    SELECT Name, Club, Jumping, Heading, Shot_Power, Finishing, Long_Shots, Curve,
Acceleration, Speed
    FROM general_info NATURAL JOIN attack_all
    WHERE Position LIKE 'Forward';

create table defence_all(Name VARCHAR(50), Club VARCHAR(30),
    Marking TINYINT, Sliding_Tackle TINYINT, Standing_Tackle TINYINT, Interceptions
TINYINT, Strength TINYINT,
    primary key(Name, Club),
    foreign key(Name, Club) references general_info(Name, Club));

create table defence(Name VARCHAR(50), Club VARCHAR(30),
    Marking TINYINT, Sliding_Tackle TINYINT, Standing_Tackle TINYINT, Interceptions
TINYINT, Strength TINYINT,
    primary key(Name, Club),
    foreign key(Name, Club) references general_info(Name, Club));

INSERT INTO defence
    SELECT Name, Club, Marking, Sliding_Tackle, Standing_Tackle, Interceptions, Strength
    FROM general_info NATURAL JOIN defence_all
    WHERE Position LIKE 'Defender';

create table goal_keeping_all(Name VARCHAR(50), Club VARCHAR(30),
    Positioning TINYINT, Diving TINYINT, Kicking TINYINT, Handling TINYINT, Reflexes
TINYINT,
    primary key(Name, Club),
    foreign key(Name, Club) references general_info(Name, Club));
```

```
create table goal_keeping(Name VARCHAR(50), Club VARCHAR(30),
   Positioning TINYINT, Diving TINYINT, Kicking TINYINT, Handling TINYINT, Reflexes
TINYINT,
   primary key(Name, Club),
   foreign key(Name, Club) references general_info(Name, Club));

INSERT INTO goal_keeping
   SELECT Name, Club, Positioning, Diving, Kicking, Handling, Reflexes
   FROM general_info NATURAL JOIN goal_keeping_all
   WHERE Position LIKE 'Goal Keeper';
```