

Final Paper – SCIAC Database

Jessie Roux

Chapman University

CPSC 408: Database Management

Rene German

May 22nd, 2021

Introduction

When it comes to competing at a high level in soccer, it is critical to be aware of your competition and all of the details regarding who you are going up against. This involves team preparation by scouting out the opposing team for key players, injuries, formations, coaches, etc. Unfortunately, it can be very difficult to get a full picture of all of this critical information in one usable format. Many teams keep individual statistics of each player, but the process is inefficient and complicated because you have to jump around and navigate multiple websites to scout players. Additionally, there is an area of need to be able to search within a collection of players for specific desires depending on what a team is scouting for. For example, a team may want to know the strongest freshmen on each team, or a list of key players that are currently out with an injury. As a current member of the Chapman women's soccer team, I can see this problem firsthand. My experience has allowed me to personally recognize the need for an application that allows a user to search players in detail and to navigate a larger quantity of key players in the SCIAC (Southern California Intercollegiate Athletic Conference) league. My competitive nature and personal connection are what jump started my passion for the project and led me to create a winning solution. My goal was to make something realistic, timely, and applicable for the future. Therefore, I have created a relational database that embodies detailed information that is specific to teams, head coaches, and players. This database keeps track of key player statistics, generates scouting reports, provides information about head coaches, and relays additional valuable information to aid in advanced and strategic scouting.

Related Work

In preparation to create this relational database, I researched existing and comparative applications to the platform I was wanting to create. Currently, the SCIAC website provides tables of individual leaders, but it only shows the top five offensive and goalie leaders. These tables lack the option to query within, to show more than five of the top players on the main display table, and to easily sort by attributes. One website that I thought had a very useful application regarding their top players in the league was that of the Women's National Soccer Team (NWSL). What stood out to me in this related work was the ability to sort by any attribute, provide the option to see more player information than just the top five league leaders, and to easily search based on the desired team. Through these related works I was able to gather user-friendly concepts that I wanted to include in my application.

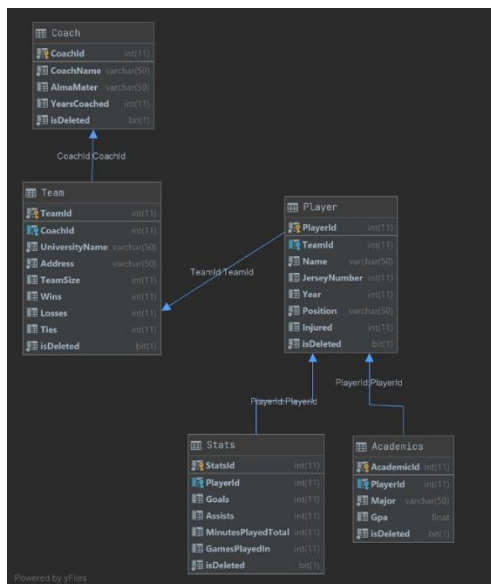
Framework

The SCIAC league database was created with MySQL Server through the Google Cloud Platform. The user interface was built using Python as a console application. In order to generate certain scouting reports and displays, Faker, Pandas and CSV were all libraries that were utilized. The framework of this application revolves around five tables and allows for basic CRUD operations. I created a coach, team, player, academic, and statistic table, all of which enforce referential integrity through primary and foreign key restraints. To ensure third normal form within my player table, I created the academic table and the statistic table, each containing a player foreign key. This ensured there are no duplicates created and eliminated any insert, update or delete anomalies. Figure 1 below highlights the overall schema of the database. All five of the tables can be seen including primary keys, foreign keys, and custom attributes. Specifically, the coach table contains a coach ID as a primary key, and attributes including name,

alma mater, and the number of years coached. Next, the team table contains a foreign key connecting to the coach table, and attributes including team ID as a primary key, name of the university, address, size of the team, and the team record. I did not put a team foreign key in the coach table in case in the future a coach was coaching two different teams: allowing for flexibility. As for the player table, it contains a foreign key connecting to the team table, also including attributes name, jersey number, current grade level, position, and injury status. Branching from the player table is the academic and statistic tables. The academic table contains a reference to the player table, and attributes of academic major and GPA. Finally, the statistic table contains a reference to the player table, and attributes such as number of goals, assists, minutes played, and game appearances. Overall, these five tables and attributes make up the schema for this SCIAC relational database.

Figure 1

Image of relational database schema.



Elements of Solution

There are many elements that come into play to create this solution to ensure efficiency, performance, and reusability. Indexes were one important element that was used to enhance performance. Indexes were created on attributes that were joined upon often in the program, such as the coach, team, and player ID's. This allowed for a quick search time when the program needed to find an attribute that was joined on frequently.

Another critical element in my solution included the implementation of a soft delete throughout all of the tables. This deletion process is identified in an "isDeleted" column in the tables. The user is only provided access to delete a player. If a player is deleted so are their academic information and statistics.

Thirdly, a main aspect in this SCIAC database is the creation process of a player. This is executed through a transaction which is inside of a stored procedure. Through the use of this transaction, commit and rollback were implemented. The creation process of a player has multiple necessary executions, so the rollback ensures that all the executions are successful or else rollback is called and the transaction is not successful.

Fourthly, in my application, the utilization of views was also very helpful. I was able to capitalize on the main aspects of flexibility and reusability that the views provided. I created a view that contained multiple joins that I used throughout my application. This allowed for dynamic access to query through the reused view depending on what was needed to be selected.

Finally, to better the performance of my application, I utilized stored procedures. I used these within the query function of my program. This was useful as I did not have to rewrite the

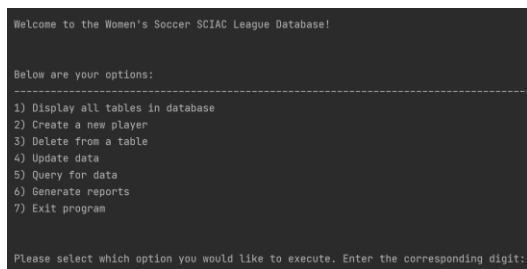
SQL statements in my program, as many of them contained joins with at least three tables. Also, stored procedures offer the opportunity to use parameters. This was helpful in being able to ask the user specifically what they wanted to query by. For example, the user can query by a specific grade level or get a list of all of the top-level players at a desired school. Finally, I implemented stored procedures when generating reports as well. This made sense as the report queries were not going to be changing.

Results

For a basic run through, my program first prompts the user questioning if they would like to display tables, create a new player, delete a player, update data, query data, generate reports, or exit the program as seen in Figure 2.

Figure 2

Screenshot of main menu options for user to choose from.



```
Welcome to the Women's Soccer SCIAC League Database!

Below are your options:
-----
1) Display all tables in database
2) Create a new player
3) Delete from a table
4) Update data
5) Query for data
6) Generate reports
7) Exit program

Please select which option you would like to execute. Enter the corresponding digit:
```

If the user selects “display tables”, all of the tables in the database are printed on the screen in a clean format. This function will always be up to date throughout the use of the application, so any changes, insertions or deletions will be reflected in the “display tables” option as seen in Figure 3.

Figure 3

Shows a portion of what the “display tables” function prints out.

	CoachId	CoachName	AlmaMater	YearsCoached
0	1	Courtney calderon	Chapman	16
1	2	Frank Marino	University of La Verne	12
2	3	Lauryn Pehanich	Cal State Fullerton	7
3	4	Cola McFeely	Olympic Development Program	29
4	5	Suzette Soboti	Skidmore College	23
5	6	Jennifer Clark	Dartmouth	8
6	7	Jennifer Scanlon	Macalester College	15
7	8	Derek Hanks	Cal State Northridge	3
8	9	Ellery Gould	Durham University	3

PLAYER TABLE							
	PlayerId	TeamID	Name	JerseyNumber	Year	Position	Injured
0	1	1	Linda Collins	23	3	Goalie	1
1	2	5	Charleen Anderson	8	3	Midfielder	0
2	3	1	Barbara Nielsen	13	3	Forward	0
3	4	4	Jasmine Alvarez	34	4	Goalie	0

Next, if the user wants to create a player, they will be prompted with three sections: soccer information, academic information and overall statistics as shown in Figure 4. With this in mind, if any of these transactions are not completed, they will be rolled back, and the player will not be created.

Figure 4

Screenshot displaying some of the process of creating a player.

```

Enter the name of the player: Jessie
Enter player's jersey number: 12
Enter player's grade level. Enter 1-4: 3
Enter the player's position: Forward
Is the player injured? Type yes or no: no

Now please enter the academic information of the player.
Enter the player's major: CompSci
Enter student's GPA: 4.0

Now please enter the stats for the player.
Enter number of goals the player has scored: |

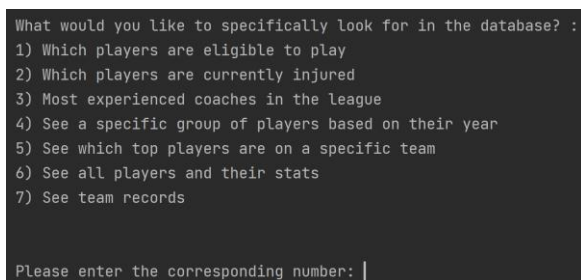
```

Next, if the user wants to delete a player, the application will perform a soft delete and update all of the tables pertaining to that player in which a deletion has been made. Furthermore, the user can query through the database. These query options are predefined based on most common and necessary queries for this database that can be seen in Figure 5. The user is able to

query for eligible players, injured players, coaches' length of experience, players based on desired academic year, top players on a specific team, all players and their statistics, and team records.

Figure 5

Screenshot of the options the user can query by.



```
What would you like to specifically look for in the database? :  
1) Which players are eligible to play  
2) Which players are currently injured  
3) Most experienced coaches in the league  
4) See a specific group of players based on their year  
5) See which top players are on a specific team  
6) See all players and their stats  
7) See team records  
  
Please enter the corresponding number: |
```

Lastly, the user is able to generate reports. These reports are generated and exported as excel files. The reports that are created are average statistics on each team, top ten goal scorers, players with maximum minutes, and the total injury count per team. In the average statistics and injury count reports, aggregates are used to compute the average and count all statistics. Finally, the user is able to exit the program with any inserts, updates, or deletes saved to the database.

Conclusion

In conclusion, this application can be very helpful to users of the SCIAC women's soccer teams, especially when it comes to prepping for games. This easy-to-use application was created to be a smoothly running platform for all users. I am looking forward to continuing to develop this application into a more interactable user interface in the near future. Through not only creating my own relational database, but also viewing my classmates, I have gained a new appreciation for data and the importance it holds in many applications.