Final Report

VFB Data Application

KC Barrett – kevin.c.barrett@vanderbilt.edu
Yue Guo – yue.guo@vanderbilt.edu
Shihan Lyu - shihan.lyu@vanderbilt.edu

Introduction

A member of the team had exposure to football operations and had access to a large historical NCAA football play-by-play dataset. He saw the project as the opportunity to develop a data application that could be complete in-game transactions of play-by-play data and post-game analytics for members of a football operations staff.

The data came from collegefootballdata.com, it consisted of over 10 seasons worth of play-by-data that

Final Implementation

Description

We are using mySQL, Python and Bootstrap templates to build a football data management and analysis website. In this data application, there are four primary functionalities: data search and Display, Data Management, Dashboard, and Operation Log. Except for the search function, all the other functions are accomplished by calling procedures, views and triggers in the database from the front end. The web application currently includes the following webpages:

Home

This page includes contact information for the development team. In the future, we believe this could be used as a login authentication page, or maybe information regarding the application's capabilities.

Data Display search and display page

This page displays dour different kinds of data: play level, play type, game score and team results. The play level data shows the information in each play, such as down, distance, yard to line and yards gained. Play type data shows all kinds of play types and whether each play type can cause scoring. These above two kinds of data come from the tables after decomposition. Game score shows how many scores did teams earn in each game and the team result displays how many games that teams won in each season. The last two kinds of data come from the views we generated from the original data.

For each data display, we also allow users to search the data by conditions. For example, when users are viewing the game result level, they can search the specific game record by home team and away team. These two conditions are logically related to OR. If the user does not fill in the search box, the default corresponding conditions are unlimited. If the user does not specify any conditions, we will display the first hundred data records of the database by default.

Data Management

In this part we have four functions for users to manage the VFB data. From our perspective, we should manage the game level data and play level data separately. Therefore, we allow users to insert a new game, update a game, delete a game and insert a play in one game.

When users want to insert a new game, the website will require them to fill in game id, home team, away team, week, season and year. Before insert, there is a procedure to check whether the game id has already existed in the game table. If id exists, it will return an error message to notify users, “Insert Failed because the game id has already existed”. Otherwise it will tell the user "Insert game successfully".

In the updating game part, the website requires users to fill in the same information as the inserting games part. But this time, the procedure will check whether the game id has already been in the game table. If not, the procedure will return an error message “This game id does not exist” to notify users. Otherwise, it will tell users “Game ID updated successfully”.

When users try to delete a game, the only thing they need to provide to the web is the game id. The website will only prompt the user that the operation is successful if the game id exists and the deletion is successful. Otherwise, an error message will be returned.

Inserting plays part is the most complex function due to the complicated football game rules. Our web application will ask users to insert all data related to the play level, including game id and drive number. Details about this feature are as follows.

1. We use game id and drive number to generate new drive id before any play data update.
2. The trigger will check the relationship between distance and yards_gained. If distance is larger than yards gained, then down will plus one. If distance is less than dance then down value needs to be set to 1.
3. The trigger will also check whether the down value is larger than 4. If it is, then trigger will swap the offense team and defense team and set down value to 1.
4. The procedure will check if the game id exists and if not, it will return correspoding error message.
5. The procedure will check the down value. If the down value is 0 and the play type is not “Extra”, then the procedure will return the corresponding error message to notify users.
6. This function will update data into three tables: drive, play, play_drive. To make sure It will not happen that only one of the three tables is inserted successfully, and the rest of the inserts fail, we use a transaction to make sure if one table insert failed, the all transaction will rollback and notify it to users.

Dashboard

In this part, we present some statistical figures for users to better analyze the performance of the team and the factors that affect the performance of the team. In this part, you can search the team and season you want, then the website can return the analysis chart.

Operation Log

The aim of this function is to record the operations happening in the database. With the operation log, we can trace back the original data and any changes on tables, which is also necessary to ensure data security and integrity. There are two logs, one is for the game table, one is for any changes related to drive data. Also, the log will display when this action happens and what specific type this action is, such as insert and update.

Database Design

R(I_D,drive_id,game_id,drive_number,play_number,offense,offense_conference,offense_score,defense,home,away,defense_conference,defense_score,period,clock,offense_timeouts,defense_timeouts,yard_line,yards_to_goal,down,distance,yards_gained,scoring,play_type,play_text,ppa,wallclock,week,season,year)

Functional dependency:
Fd1: Drive id, play number ® Offense, year, Defense, game_id,home,away,week,season,year, drive_number,period,offense_score,defense_score,offense_timeouts,defense_timeouts,wallclock,drive_number
Fd2: Offense, year® offense_conference
Fd3: Defense, year ® defense_conference
Fd4: Drive id ® game_id,home,away,week,season,year,drive_number
Fd5: Play_type ® scoring
Fd6: game_id ® game_id,home,away,week,season,year
Step 1
According to FD 1, we can split the R into two parts:
Play(Drive id, play number ,Offense, year, Defense, game_id,home,away,week,season,year, drive_number,period,offense_score,defense_score,offense_timeouts,defense_timeouts,wallclock,drive_number, offense_conference, defense_conference)

R(I_D, drive_id ,clock,yard_line,yards_to_goal,down,distance,yards_gained,
scoring,play_type,play_text,ppa)

Step 2

According to FD2 and FD3, we can split play into 3 parts:
Team_offense(Offense, year, offense_conference)
Team_offense(Defense,year, defense_conference)
Play(Drive id, play number,Offense, year, Defense, game_id,home,away,week,season,year,
drive_number,period,offense_score,defense_score,offense_timeouts,defense_timeouts,wallclock,drive_n
umber)

Step 3

According to FD4, we can split play into two parts:
Drive(Drive id, game_id,home,away,week,season,year, drive_number)
Play(Drive_id,play_number,Offense,year,Defense,period,offense_score,defense_score,offense_timeouts,
defense_timeouts,wallclock,drive_number)

Step 4

Acoording to FD5, we can split R into two parts:
R(I_D, drive_id ,clock,yard_line,yards_to_goal,down,distance,yards_gained, play_type,play_text,ppa)
Playtype(play_type, scoring)

Step 5

According to FD6, we can split drive into two parts:
Drive(Drive id, game_id, drive_number)
Game(game_id,home,away,week,season,year,)

The data was initially loaded into the database as a table called `all_plays`. This was the mega table that was composed of thirty fields and 1.56 million plays. These data fields are provided in the appendix, table#. Right away we knew that there was an inherent hierarchy within the play-by-play data that represented a complex series of relationships.
Games occurred between two distinct opponents, where one team was `home` and the other team was `away`. Each team could be a member to a single conference, but it was possible for a team to not be a part of a conference. While it was possible for teams to change conferences between seasons, the change never occurred within a season.

Within each game, the teams would alternate between `offense` and `defense`. We noticed that the `offense` and `defense` would remain constant over a series of plays and would only change under specific conditions that resulted in a change of possession. We saw that these plays were uniquely identified with the drive_id, so we went ahead and called them drives. The play level data within a drive would detail the majority of the data the fields that would be needed for future analysis. Each play referenced the offense, defense, down, distance, yardline, yards_to_goal, clock, period, play_type, play_text, yards_gained, and whether it was scoring. Using this logic, we set about decomposing the data into tables in a manner that would be lossless.

The final database design has been provided in Table 1: Database Table Decomposition on page

Table 1: Database Table Decomposition

Table Name	Description	Data Fields	Primary Key	Foreign Key & Reference	Relationship
team_offense	Record the offense team, the year game happens and the cofference	offense, year, offense_conference	offense , year	None	Every offense teams should exists in this table
team_defense	Record the defense team, the year game happens and the cofference	defense, year, defense_conference	defense , year	None	Every defense teams should exists in this table
game	Record every game and related information	game_id, home, away, season, week, year.	Game_id	None	Have game first then we can have drive and play.
drive	Record the relationship between drive id and game.	drive_id, game id, drive number.	Drive_id	Game_id related with game table	Drives belong to games. One drive can have multiple plays.
playtype	Record all the play types in a football game and whether they can cause scoring.	play_type, scoring	Play_type	None	All the play types need to come from this table.
play	Record all the plays happened in games	drive_id, play_number, period, offense_score, defense_score, offense_timeouts, defense_timeouts, wallclock,offense,defense	Drive_id, play_number	Drive_id related with the drive table, offense, defense are related with the team offense and defense table	There can be a lot of play in one drive. In each play
play_drive	There is dirty data in our dataset. To make sure that the table can satisfy 3CNF, we split this table from the play table.	I_D, drive_id, play_number, clock, yard_line, yards_to_goal, down,distance, yards_gained, play_type, play_text, ppa	I_D		
drive_diary	This table records the operations happened related to drive	action_id,game_id,home,away,week,season,year,action_time,action	action_id	None	After insert on three tables which have drive id
game_diary	This table records the operations happened on	action_id,drive_id,play_number,action_time,action	action_id	None	After insert, update and delete on game table

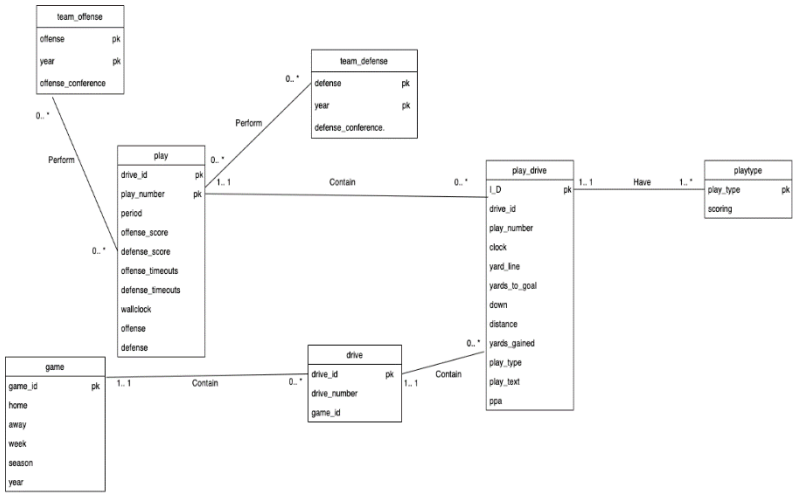
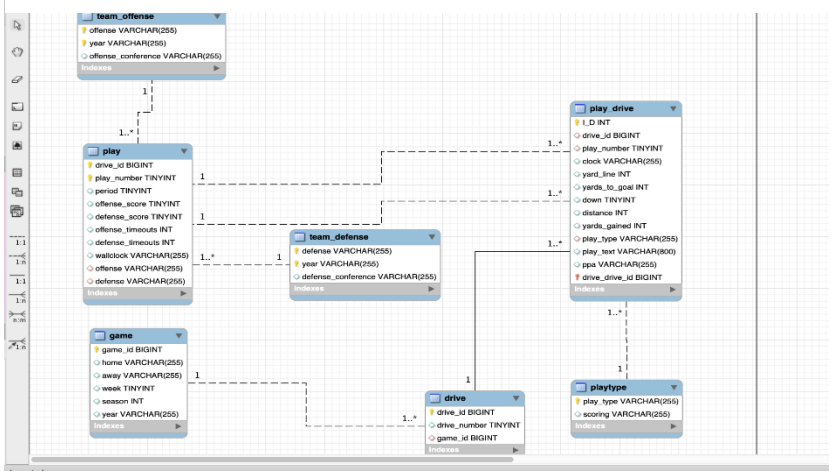


*Figure SEQ Figure * ARABIC 2: Physical Model*



Testing

All these test data can be successfully operated on the website. These do not include test error messages and scenarios that prompt errors

Display	Search play	Drive id: 31244000901	1
	Search play type	Play type: End of Game;Scoring: True	1 or 2(choose whatever you want)
	Search game score	Home: Georgia Tech; Away: UC Davis; Season: 2011	1 or 2 or 3 (choose whatever you want)
	Search team results	Season: 2011; Team: Georgia Tech	1 or 2 (choose whatever you want)
Management	Insert a new game	Game id: 1; Home: Testhome; Away: Testaway; Week: 1; Season:2020; Year: 2020	6
	Insert a new play	Game id:1; Drive number:1; Offense: Air Force; Offense Score: 1; Defense: Abilene Christian ; Defense Score: 1; Play number: 1; Clock: test; Yard Line: 1; Yard to Goal: 2; Yards gained:1; Down: 2; Distance:4 Period:1; Play Type: Blocked Punt Touchdown; Play Text: test;	16
	Update a game	Game id: 1 Home: Testhome1; Away: Testaway1 Week: 1 Season:2020; Year: 2020	6
	Delete a game	Game id: 1	1
Dashboard	Search figures	Team: Rice; Season: 2017	2
Action Log	Game operation log		0
	Drive operation log		0

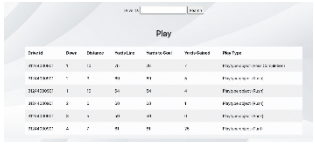
Functionality

The

Functionality

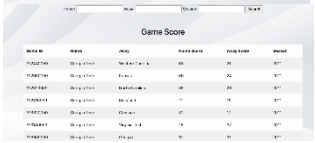
The VFB Alpha release includes the following instructions for using the application. Screenshots have been provided.

Display Search Results



Play Data: The User inputs this id: 31244000901 into the search box and clicks the search button. Then he will get the result as follows.

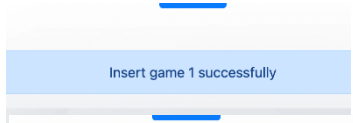
Play Yype: Click on the “Play Type” to jump to another search page. The User inputs value: End of Game, into play type search box and clicks the search button. Then the app will return the result as follows.



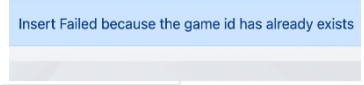
Game Score: Click on the “Game Score” to jump to another search page. The User inputs value: Georgia Tech, into Home search box, and inputs value:2011, into Season box and clicks the search button. Then the app will return the result as follows.

Team Results: Click on the “Team Results” to jump to another search page. The User inputs value: Georgia Tech, into Team search box, and inputs value:2011, into Season box and clicks the search button.Then the app will return the result as follows.

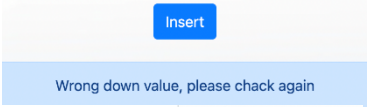
Management (Steps to Validate Testing Data)



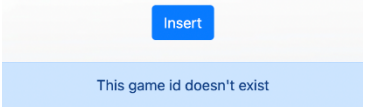
Insert Game Successfully: Click on the “insert a new game” button. The user inserts the sample data in the testing data section. Then the app will return the result as follows



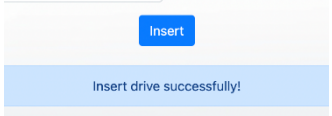
Insert Game Failed:
The user inserts the sample data in the testing data section again. Then the app will return the result as follows



Insert new play failed 1: Click on the “insert a new play” button. The user inserts the sample data in the testing data section, but change the game id to 0.Then the app will return the result as follows



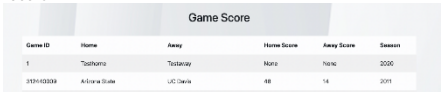
Insert new play failed 2: The user inserts the sample data in the testing data section, but change the ‘down’ value to 0.Then the app will return the result as follows.



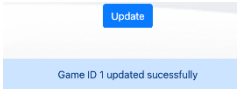
Insert new play successfully: The user inserts the sample data in the testing data section.Then the app will return the result as follows.

Check whether the data interaction is successful 1: Click on the “Display” in the navbar. The user inserts drive id as 101. Then the app will return the following: *Please note that This occurs because drive id = game id *100 + drive number. Therefore, the new drive id is 101. And also the sample data has distance larger than yards gained, Therefore the down value should be added 1: 2+1 = 3*

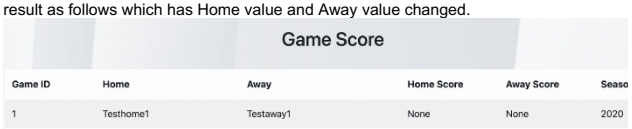
Check whether the data interaction is successful 2 : Click on the “Game Score” button. Then the app will show the result as follows which has game id 1 record.



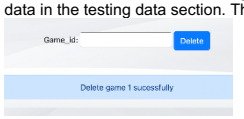
Update successfully: Click on the “Management” in the navbar and click on “update game” button. The user inserts sample data in the testing data section. Then the app will show the result as follows which has game id 1 record.



Check Whether the Data Interaction is Successful 3: Click on the “Display” in the navbar and click on the “Game Score” button. Then the app will show the result as follows which has Home value and Away value changed.

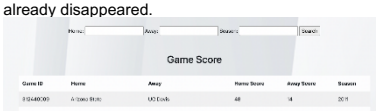


Delete Game: Click on the Management” in the navbar. The user inserts sample data in the testing data section. Then the app will show the result as follows



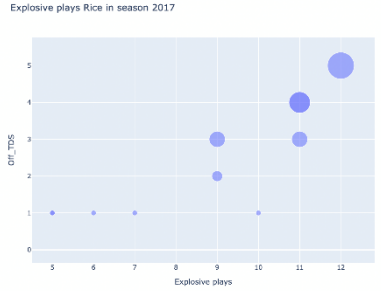
Check whether the data interaction is successful 4: Click on the “Display” in the navbar and click on the “Play” button. The user inserts the drive id value as 1 in the search box and click search button. Then the app will show the result as follows, which game id 1 has already disappeared.

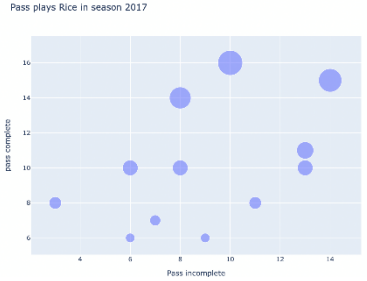
Check whether the data interaction is successful 5: Click on the “Game Score” button. Then the app will show the result as follows, which game id 1 has already disappeared.



Dashboard

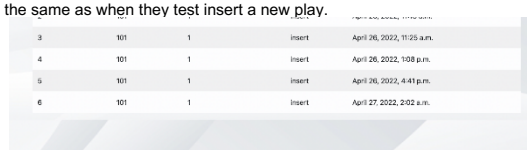
Dashboard search: Click on the “Display” in the navbar. The user inserts sample data in the testing section. Then the app will show the result as follows.





Action Log

Click on the “Display” in the navbar and click on the “Drive Operation log” button. Then the app will show the result as follows. Users can check if the action time is the same as when they test insert a new play.



Click on the “Game Operation log” button. Then the app will show the result as follows. Users can check if the action time is the same as when they test insert, update or delete the game id 1

Summary Discussion

Status and stopping point

The present state demonstrates that a large historical dataset of play-by-play data can be decomposed in a manner will facilitate

Challenging part

For our project, the challenge is that the rules of football games are very complicated, and only one student in our group is familiar with football games. We need to apply our application to the actual football game, for example, we need to write a lot of triggers and procedures to verify whether the data entered by the user is correct.

We have probably had no less than five sprint meetings. Students who understand the data at the meeting will spend a lot of time explaining the meaning of each data to us to ensure that we understand the relationship between the data. At the same time, we will discuss the points where no function needs a checksum check in the backend to ensure that we can fully meet the requirements of football data

Division of labor

Barrett, Kc:

1. Data decomposition
2. Database creating, data loading.
3. Statistical views and procedures (2 views and 2 procedures in total)
4. Presentation Powerpoint

Guo, Yue：

1. Table creating after decomposition, data inserting
2. Front end coding and designing
3. Management part procedures (4 in total)
4. Management part and operation log part triggers(5 in total)

Lyu, Shihan

1. UML
2. Insert procedure (1 in total)

Explanation:

KC has a very good understanding of football games and related data, and is able to proficiently calculate the relationship between various variables, so he undertakes the work of data decomposition and statistical view and procedure generation. At the same time, KC's presentation experience is rich. Yue has previous product working experience and is familiar with SQL code. Therefore, she undertakes the generation of front-end code and some SQL features. Shihan is familiar with UML and Shihan took the initiative to undertake the UML and the rest of the code.