StatAP

Custom statistics for the NFL

Bharat Gupta, Robert Mayer, Aaron Neyer, and Diego Waxemberg

May 4, 2013

Final Report

Contents

List of Figures

1 Introduction

1.1 Abstract

The National Football League (NFL) is one of the most popular and profitable professional sports leagues in the world. The NFLs popularity has led to a surge in the collection of game data and analysis of this data. Football, due to its innate complexity, comprises of hundreds of variables for which data can be collected. These variables include offensive yards, touchdowns scored, passes intercepted, and much more. Furthermore, this data can be associated with individual players, coaches, and the team as whole which creates new ways in which the data can be interpreted. The implications of these statistics have created an avenue for computer applications (largely relying on the internet) where users can play Fantasy Football, predict and bet on games, view games live, and just plainly view the data itself. Furthermore, this data has led to the creation of new statistical measures to characterize players like the Quarterback Rating (QBR) which evaluates the proficiency of a Quarterback, the player that executes the offensive plays.

1.2 Problem Definition

The NFL provides access to most of the data it collects online for free. Various 3rd party companies also provide this data, and provide access to special data not easily made available by the NFL. While this data along with statistical measures are made available to the public, there is no major platform available on which users can extract custom statistics. Often on TV, a sports commentator will provide an elaborate statistic (e.g. player x has won the last 5 games against team y when the team of player x has 300 or more offensive yards) which will give more insight into the game, and interests fans in general. Currently, most fans rely on commentators to provide them with such insights as no major platform exists where they can resolve custom statistics.

1.3 Application Overview

StatAP will consist of a newly created database which will consist of a subset of the data provided by the NFL as it is unrealistic to include all of the data. Other data will be incorporated into this database including weather conditions during the game and location of the game. The goal is to build a tool where users can generate custom statistics which they may themselves declare to find insightful. The application will center around querying the custom database to find useful and relevant statistics of which there are too many to list on any single page. The data can be extracted online using data mining tools. There are websites that provide data in spreadsheet format as well. Data not associated with the NFL can be extracted easily as well. The application can be expanded in many ways. The most obvious is just to create more and more ways to query the database. Live data extraction can also be incorporated into the application. Furthermore, a model that can potentially predict games can be created and improved using our database by the principles of AI.

2 Application Requirements Specifications

Go over the application requirements...

2.1 Homepage

The home page will mainly be used to navigate to either the Players or Teams section of the website which will be the two main sections. The home page may also display some interesting statistics, and explain how these statistics were arrived at to give users an idea on how to use the application. However, the home page itself will not have much other functionality.

2.2 Players View

The players page will show a list of every player, sorted by some default setting, such as alphabetical. On the top, there will be a number of ways to filter. There will be a select box and a search box, allowing you to do a text search, which would primarily be used to search by player name. There would be other filters, such as selecting only players on a certain team, or selecting only quarterbacks who have thrown for more than 3000 yards in 2011. The list would show the results in a given amount at a time(defaulting to 30), and then paginate the result

1. Positions(QB, RB, etc...)
2. Search by player name
3. Filter by statistics
4. Filter by team

2.3 Teams View

B) The teams page will initially start out as a view of all of the teams sorted alphabetically with a search and filter bar at the top. From a drop down box the user can choose to either keep the teams sorted alphabetically or by wconference and division. The order that divisions will show is AFC East, AFC North, AFC South, AFC West, NFC East, NFC North, NFC South, NFC West. The search bar will be located in the top right hand side of the page allowing the user to type in a specific team name or city to quickly access team statistics they are looking for. Some examples of the filters that could be entered include: only showing teams with over 60 wins in the last 10 years, teams that have quarterbacks with a rating above 100, etc.

1. Conference, division
2. Filter by statistics
3. Search by team name

3 Database Requirement

Put in a description

3.1 Entities

3.1.1 Player

This table will hold all static values relevant to a certain player. This includes things like, the players position, name, id, etc. We will use this table to get information about a player that does not relate to their game stats.

3.1.2 Teams

The teams table is used to identify a team by its id or name and get information regarding that team.

3.1.3 Game

The games entity will contain all information relevant to a specific game, such as when it took place. This will be used to determine teams' win statistics. Since this entity will have an attribute for the home team and away team, we will be able to determine how much home-field advantage affects each team. We can also compare the weather of a certain game with its outcome.

3.2 Relations

3.2.1 Player Team

This relation determines the team a particular played/plays for wit specified begin and end date.

3.2.2 Player Game

This relation specifies if a player played in a particular game.

4 Integrity Constraints

4.0.3 Description

List any integrity constraints in general on the entities and relationships as a whole. Explain how you intend to enforce them; i.e., are you going to build enforcement mechanisms by: specifying sql constraints or triggers, applying database dependency theory, or applying exernal integrity enforcement. You must explicitly specify general constraints, triggers, and stored/external procedures.

4.0.4 Breakdown

At this stage of the project, it does not appear that the user will be allowed to modify the database itself. The database will be populated with freely available data from the internet. The user will be able to use complex queries to generate customized statistics which will not result in any change in the database. It is anticipated that users will restricted to what kind of input they can use i.e a user will not be able to enter any custom input but will be restricted. For example for a statistics filter the user will be limited from entering 1 to 500 only, due to the fact that this encompasses all possible values for a stat and prevents the user from using malicious input. This is why integrity constraints will not be a huge factor. The integrity constraints that may come into play are listed below:

1. Player name, stats, etc cannot be NULL
2. player id team id as primary ke
3. team id player id

5 Queries

5.1 5 Main Queries

5.1.1 Find the average number of yards Chris Johnson has while playing in a game for the Titans between the years 2008 and 2010.

SQL

The first item

RA

$ChrisGames \leftarrow (\sigma_{year=2008 \vee year=2009 \vee year=2010}((\sigma_{team_name=Titans}((\sigma_{player_name=ChrisJohnson}(Game \bowtie PlayersGame \bowtie Player)) \bowtie Team))))$

$ChrisGames_{average(\sigma(yardage))}$

Yardage can equal rushing_yards, passing_yards, receiving_yards, punt_return_yards, or kick_return_yards.

TRC

$\{Average(t^{(1)}|(\exists g)(Games(g) \wedge (\exists p)(Player(p) \wedge (\exists T)(Team(T) \wedge (\exists py)(PlayerYear(py) \wedge (\exists pg)(PlayerGame(pg) \wedge (\exists ps)(PlayerStats(ps)[1] = ps.rushing\_yards \wedge p.name = "ChrisJohnson" \wedge p.pid = py.pid \wedge (py.year = "2008" \vee py.year = "2009" \vee py.year = "2010") \wedge p.pid = pg.pid \wedge pg.tid = T.tid \wedge T.team\_name = "Titans" \wedge pg.player\_stats\_id = ps.player\_stats\_id))))))$

5.1.2 Find the teams in any specific year that had a player get greater than or equal to a certain number of yards.

SQL

The first item

RA

$PositionYear \leftarrow \sigma_{year=input\_year}((\sigma_{position=input\_position}Player) \bowtie PlayersYear)$

$Yards \leftarrow \sigma_{passing\_yards \geq numyards}((PositionYear \bowtie PlayersGame) \bowtie PlayerStats)$

$\Pi_{team\_name}(Yards \bowtie Team)$

TRC

$\{t^{(2)}|(\exists p)(Player(p) \wedge (\exists T)(Team(T) \wedge (\exists py)(PlayerYear(py) \wedge (\exists pg)(PlayerGame(pg) \wedge (\exists ps)(PlayerStats(ps) \wedge t[1] = T.name \wedge t[2] = T.tid \wedge p.pid = py.pid \wedge p.position = "input\_position" \wedge py.year = "input\_year" \wedge p.pid = pg.pid \wedge pg.player\_stat\_id = ps.player\_stat\_id \wedge ps.yards \geq numyards \wedge pg.tid = T.tid)))))\}$

5.1.3 Find the city that has the most number of away team wins in a given year.

SQL

The first item

RA

$AwayGames \leftarrow \sigma_{year=input\_year}((Team \bowtie_{team\_id=home\_team\_id} Game) \bowtie TeamYear)$
$\Pi_{city}(AwayGames \bowtie TeamStats)_{max(\sigma(losses))}$

TRC

$\{Max(t^{(1)}) | \exists T(Team(T) \wedge (\exists g)(Game(g) \wedge (\exists ty)(TeamYear(ty) \wedge (\exists ts)(TeamStats(ts) \wedge$
$t[1] = ts.losses \wedge$
$T.tid = G.home\_team\_id \wedge T.tid = ty.tid \wedge ty.year = "input\_year" \wedge ty.team\_stats\_id = ts.team\_stats\_id)))))\}$
$\{Max(u^{(1)}) | (\exists T)(Team(T) \wedge t[1] = T.city \wedge (\exists g)(Game(g) \wedge (\exists ty)(TeamYear(ty) \wedge (\exists ts)(TeamStats(ts) \wedge$

$T.tid = G.home\_team\_id \wedge T.tid = ty.tid \wedge ty.year = "input\_year" \wedge ty.team\_stats\_id = ts.team\_stats\_id \wedge ts.losses = t[1])))))\}$

5.1.4 Find the team with the most offensive yards in a given year.

SQL

The first item

RA

$YearTeam \leftarrow (\sigma_{year}(Team \bowtie TeamYear))$
$TotalYards \leftarrow (YearTeam \bowtie TeamStats)_{sum(offensive\_yards)}$
$\Pi_{team\_name}(TotalYards_{max(\sigma(offensive_yards))})$

TRC

$\{Max(t^{(1)}) | (\exists ts)(TeamStats(ts) \wedge t[1] = ts.offensive\_yards \wedge (\exists T)(Team(T) \wedge (\exists ty)(TeamYear(ty) \wedge$

$T.tid = ty.tid \wedge ty.team\_stats\_id = ts.team\_stats\_id)))\}$
$\{u^{(2)} | (\exists T)(Team(T) \wedge u[1] = T.name \wedge u[2] = T.tid \wedge (\exists ty)(TeamYear(ty) \wedge (\exists ts)(TeamStats(ts) \wedge$
$T.tid = ty.tid \wedge ty.team\_stats\_id = ts.team\_stats\_id \wedge ts.offensive\_yards = t[1]))))\}$

5.1.5 Find a teams average number of a specific type of yards for a given position in a given year.

SQL

The first item

RA

$PlayerPosition \leftarrow (\sigma_{year=input\_year}(\sigma_{position=input\_position}(Player)) \bowtie PlayerYear)$
$PositionStats \leftarrow (PlayerPosition \bowtie PlayersGame \bowtie PlayerStats)$
$\Pi_{yardage}((PositionStats \bowtie Team)_{average(\sigma(yardage))})$
Yardage can equal rushing_yards, passing_yards, receiving_yards, punt_return_yards, or kick_return_yards.

TRC

$\{Average(t^{(1)}) | (\exists ps)(PlayerStats(ps) \wedge t[1] = ps.yardage \wedge (\exists p)(Player(p) \wedge (\exists py)(PlayerYear(py) \wedge$

$p.position = "input\_position" \wedge p.pid = py.pid \wedge py.year = "input\_year" \wedge (\exists T)(Team(T) \wedge$
$(\exists pg)(PlayerGame(pg) \wedge$
$p.pid = pg.pid \wedge pg.tid = T.tid \wedge T.name = "input\_name" \wedge pg.player\_stats\_id = ps.player\_stats\_id))))))\}$

5.1.6 .

SQL

The first item

RA

$Games\_y \leftarrow \Pi_{Game\_id}(\sigma_{name="input\_name"}Team \bowtie_{team\_id} \sigma_{year="input\_year"}Games)$
$Team\_Players\_y \leftarrow \Pi_{player\_id}(\sigma_{start\leq"input\_year"\wedge end\geq"input\_year"}PlayerTeam \bowtie_{team_id} \sigma_{name="input\_name"}Tea$
$(Team\_Players\_y \bowtie_{player\_id} PlayerGame)/Games\_y$

TRC

$\{t^{(2)}|(\exists p)(Player(p)\wedge(t[1] = p.pid\wedge r[2] = p.name)\wedge(\exists T)(Team(T)\wedge T.name = "input\_name"\wedge$

$(\exists pt)(PlayerTeam(pt) \wedge pt.pid = p.pid \wedge pt.tid = T.tid \wedge pt.startdate \leq input\_year \wedge$
$pt.enddate \geq input\_year \wedge$
$(\forall g)(Game(g)\wedge g.year = input\_year\wedge(g.home\_team\_id = t.tid\vee g.away\_team\_id = t.tid) \rightarrow$
$(\exists pg)(PlayerGame(pg) \wedge$
$pg.pid = p.pid \wedge pg.gid = g.gid \wedge pg.tid = t.tid)))))\}$

5.1.7 Other Queries

1. Determining a teams win/loss record over a given period of time.
2. Determining a teams record against an opposing team over a given period of time.
3. Determining a teams record without a given player over a given period of time.
4. Calculating a teams offensive/defensive yardage with or without a given player.
5. Determining a teams performance based upon the weather conditions.
6. Determining a teams performance at a given location/stadium.
7. Finding out a teams best performance offensively or defensively in a given period of time.
8. Selecting all players who match a given statistics filter (e.g select a player with x or more yards of offense).
9. Find all players who have had x or more wins against team A.
10. Rank all players based upon a given stat (e.g rushing yardage).
11. Find two players that have combined for the most offensive yards in a given period of time.
12. Determine a players career record against a given team.
13. Determine a players career stats against a given team.
14. Determine a players win/loss ratio if he generates x or more yards against a given team.

Figure 1: Example

6 Entity Relational Model

6.1 Description

Specify the properties of each attribute (i.e., key, composite/simple, singlevalued/multivalued, derived, incomplete with different nulls, roles, weak/strong entity type, etc.). Specify the relationships and their attributes. List the properties of each relationship (i.e., degree of the relationship, cardinality ratio constraints (11, 1N, NM), key constraints, participation constraints (total, partial), other applicationspecific constraints, etc.).

6.2 StatAP ER Diagram

7 Relational Model

Each entity should map to a (entity) relation, and each relationship should be accounted for (either represented in an entity relation or as a separate relation). Each entity/relationship attribute should be accounted for in the transformation. List for each relation the primary/candidate keys, foreign keys. Specify the entity and referential integrity constraints, and discuss why they are satisfied for each relation. Explicitly specify each relation using CREATE TABLE commands of SQL

8 Database Implementation

8.1 Description

Summarize the main components of your code here with proper explanations. If you are using SQLServer, present (using screen dumps of the query analyzer output,) the actual logical query trees for the five most difficult queries. Discuss any problems encountered and how you have solved them. If some of the problems require you to extensively redesign everything, you may choose to point them out, and not implement them. If some of your stored procedures are too elaborate, scale down your design, explain your decisions, and implement the scaled down version

8.2 Process

We are using MySQL for our DBMS. We set up the open source DBMS on your respective machines without any issues by using tools like YUM and APT-GET on the linux terminal. We also downloaded and installed MYSQL workbench in order to manage our database. As we are building a Ruby on Rails application, we created the database schema using Rails and used a Rails gem Railtoady to generate our relational diagram. However, we were not able to generate a nice ER diagram but we will have one done for the final report. Most of the commands we executed were through the command line interface. A screenshot of our DBMS is attached below.

8.3 SQL

9 Relational Database Design

9.1 Description

More specifically, define your functional dependencies, and find the minimal set of f.d.s. Are all your relations in BCNF or 3NF? If not, apply the algorithms you have seen in the class to decompose and make them BCNF/3NF. Note that your decompositions should be lossless and dependency preserving. 1) Game
Keys:
$gid \rightarrow \{date, year, week, home\_team\_id, away\_team\_id, home\_team\_stat\_id, away\_team\_stat\_id, result\}$
$home\_team\_stat\_id \rightarrow \{determineseverything\}$
$away\_team\_stat\_id \rightarrow \{determineseverything\}$

2) Player
Keys:
$pid \rightarrow \{position, name, url\}$
$url \rightarrow \{determineseverything\}$

3) Team
Keys:
$tid \rightarrow \{name, city, active\}$
$name \rightarrow \{determineseverything\}$

4) PlayerTeam
Keys:
$\{\}emptyset$

5) PlayerGame
Keys:
$\{\}emptyset$

6) PlayerStats
Keys:
$player\_stats\_id \rightarrow \{determineseverything\}$

7) TeamStats
Keys:
$team\_stats\_id \rightarrow \{determineseverything\}$

8) PlayersYear
Keys:
$player\_year\_id \rightarrow \{player\_id, year, player\_stats\_id\}$

9) TeamYear
Keys:

$team\_year\_id \rightarrow \{team\_id, player\_stats\_id, team\_stats\_id, year\}$

10 Application Implementation

10.1 Overview

We built this application using the Ruby on Rails framework which integrates several components together for quick development and deployment of a scalable web application.

10.2 Installation Manual

In order to install and run our application you must install the ruby and rails development environment on your machine. This environment is open source and compatible with Windows/-Mac/Linux. We recommend using the Ruby Version Control Manager for installation. [1] Once you have installed the Ruby on Rails environment on your machine, please follow the following steps to run our application.

1. Visit our Github repository, and download the source code.[2]
2. Open up a terminal and run **bundle install** after going inside the eecs-341 directory (note that rails will automatically detect that this is a rails application).
3. Now run **rake db:create** followed by **rake db:migrate**.
4. Start the server by running **rails s**.
5. Finally open a web browser and visit **localhost:3000**. [3]

[1]You can get rails at: `http://guides.rubyonrails.org/getting_started.html`
[2]Github url: `https://github.com/Aaronneyer/eecs-341`
[3]Please note rails uses port 3000 by default, you can specify the port with -p.

11 Appendix

11.1 User Manual

Figure 2: Example

11.2 Programmers Manual