CS 4750 Project—Football Database

Yong Jae Kwon(yk7da), Taehyun Kim(tk9at), Andrew Yang(ahy9ng)

Department of Computer Science, University of Virginia

Abstract: This project's main goal is to create a database and corresponding website that contains the data of all football players in the 6 football teams in Virginia.

1. Project Information.

1. Introduction

Our project's goal is to create a database for a college football website. The users of this website will be able to search different football players in the 6 NCAA-1 level football teams in Virginia and get the match data of those football teams. Each player information stores the player's height, weight, main class, address and team. Each match information stores the date, the participating teams, scores for each team, the passing leader, the rushing leader and the receiving leader. Each user will have an account that stores information of their favorite players. The administrator will have a separate account that can insert and delete players from the players table, which is linked with other school-specific and position-specific tables.

2. Requirements

Users should be able to create, sign in and sign out from accounts. Accounts will only contain the user's username, password, and their favorite player list. The information of individual football players will be stored in the Players table in the football database. The details for each player includes their number, position, class, height, weight, hometown, state, team and their ID (which was generated respective to their team and number). These details will be stored in the main “Players” table and the tables corresponding to the player’s school and team position. The tables have constraints to detect if a player is truly being in a certain school/position. The information of individual matches, which will be stored in the “match_2016 table”, including details of the two teams, the final score, the date and time and leading players. Users will be able to search certain players either by their names, or their team, or their position. They will also be able to insert certain players in their favorite players list, and to delete items from the favorites list. The user can also export the Players table in JSON form. They can also look at the match information for the 6 teams in Virginia, and see the list of local players that has either been the passing leader, the rushing leader, or the receiving leader for a match. The admin user will be able to insert/delete players from the Players list, which, using triggers, will also change the tables corresponding to the player.

Security is handled so that it is safe against SQL injection attacks. There is a separate user that will be the only user which will be reading that login table. As mentioned above, there is only one admin that can do the insert/delete operations on the entire database, and the rest of the users can only select some tables or insert/delete the favorites table.

2. Design Process

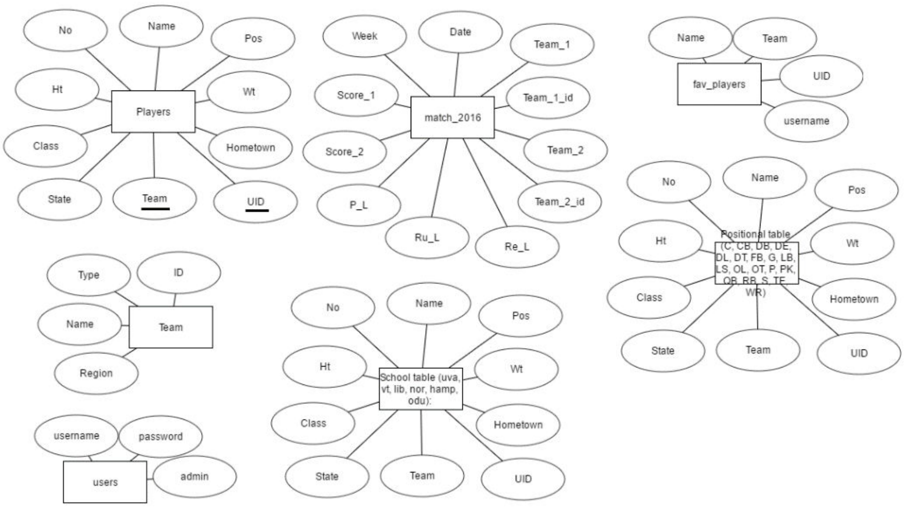
1. Design Decisions

To make things easy for users to access and acquire information, we decided to use website as the interface application. While we didn't used data encryption, for security we separated the data, code and prepared for possible SQL injection threats. We also granted different types of privileges for different users.

Most of the database is consisted of player data. The "Players" table is consisted of the information of football players, including their current team and main position. There exist other tables that includes information of football players in a certain school or in a certain position (Example: table "UVA" contains all players in Virginia Cavaliers, table "QB" contains all quarterbacks). The "match_2016" table, unlike the other tables, consists of the match data between two teams, including the "leaders" for each match. The "users" table contains the username and the password for each users using the system, and the "fav_players" table contains the name, team, and UID for the players chosen by the currently logged user.

2. E-R Diagram.

Because there are a lot of tables and attributes for each table, the tables with same structures (i.e. school table and positional table) was grouped up. The E-R Diagram is shown below.



3. Database Schema

Players:

`No` varchar (2) CHARACTER SET utf8 DEFAULT NULL,

`Name` varchar (23) CHARACTER SET utf8 DEFAULT NULL,

`Pos` varchar (2) CHARACTER SET utf8 DEFAULT NULL,

`Ht` varchar (6) CHARACTER SET utf8 DEFAULT NULL,

`Wt` int (11) DEFAULT NULL,

`Class` varchar (2) CHARACTER SET utf8 DEFAULT NULL,

`Hometown` varchar (21) CHARACTER SET utf8 DEFAULT NULL,

`State` varchar (2) CHARACTER SET utf8 DEFAULT NULL,

`Team` varchar (3) CHARACTER SET utf8 NOT NULL DEFAULT ,

`UID` varchar (5) NOT NULL DEFAULT

```
   PRIMARY KEY (`Team`,`UID`)

   CREATE TRIGGER `Players_trigger` AFTER INSERT ON `Players`

     FOR EACH ROW BEGIN

   IF (new.Team = ‘player_team)

   THEN

            INSERT INTO ` player_team ` (`No`, `Name`, `Pos`, `Ht`, `Wt`, `Class`,
`Hometown`, `State`, `Team`, `UID`) VALUES ( new.No, new.Name, new.Pos,
new.Ht, new.Wt, new.Class, new.Hometown , new.State , new.Team, new.UID );

   END IF;

   IF (new.Pos = ‘player_pos’)

   THEN

            INSERT INTO `player_pos` (`No`, `Name`, `Pos`, `Ht`, `Wt`, `Class`,
`Hometown`, `State`, `Team`, `UID`) VALUES ( new.No, new.Name, new.Pos,
new.Ht, new.Wt, new.Class, new.Hometown , new.State , new.Team, new.UID );

   END IF;

   END
```

School table (uva, vt, lib, nor, hamp, odu):

```
    `No` varchar (2) CHARACTER SET utf8 DEFAULT NULL,

    `Name` varchar (23) CHARACTER SET utf8 DEFAULT NULL,

    `Pos` varchar (2) CHARACTER SET utf8 DEFAULT NULL,

    `Ht` varchar (6) CHARACTER SET utf8 DEFAULT NULL,

    `Wt` int (11) DEFAULT NULL,

    `Class` varchar (2) CHARACTER SET utf8 DEFAULT NULL,

    `Hometown` varchar (21) CHARACTER SET utf8 DEFAULT NULL,
    `State` varchar (2) CHARACTER SET utf8 DEFAULT NULL,

    `Team` varchar (3) CHARACTER SET utf8 NOT NULL DEFAULT ,

    `UID` varchar (5) NOT NULL DEFAULT

  PRIMARY KEY (`Team`,`UID`)

  ADD CONSTRAINT `FK_PLAYER_(school_name)` FOREIGN KEY (`Team`,
`UID`) REFERENCES `Players` (`Team`, `UID`) ON DELETE CASCADE ON
UPDATE CASCADE;
```

Positional table (C, CB, DB, DE, DL, DT, FB, G, LB, LS, OL, OT, P, PK, QB, RB, S, TE, WR):

```
    `No` varchar (2) CHARACTER SET utf8 DEFAULT NULL,

    `Name` varchar (23) CHARACTER SET utf8 DEFAULT NULL,

    `Pos` varchar (2) CHARACTER SET utf8 DEFAULT NULL,

    `Ht` varchar (6) CHARACTER SET utf8 DEFAULT NULL,

    `Wt` int (11) DEFAULT NULL,

    `Class` varchar (2) CHARACTER SET utf8 DEFAULT NULL,

    `Hometown` varchar (21) CHARACTER SET utf8 DEFAULT NULL,

    `State` varchar (2) CHARACTER SET utf8 DEFAULT NULL,

    `Team` varchar (3) CHARACTER SET utf8 NOT NULL DEFAULT ,

    `UID` varchar (5) NOT NULL DEFAULT

  PRIMARY KEY (`Team`,`UID`)

  ADD CONSTRAINT `FK_PLAYER_(pos_name)` FOREIGN KEY (`Team`,
`UID`) REFERENCES `Players` (`Team`, `UID`) ON DELETE CASCADE ON
UPDATE CASCADE;
```

Users:

```
      `username` varchar (16) CHARACTER SET utf8 NOT NULL DEFAULT,
      `password` varchar (16) CHARACTER SET utf8 NOT NULL DEFAULT,
      `admin` tinyint(1) NOT NULL DEFAULT
```

Fav_players:

```
      `Name` varchar (18) CHARACTER SET utf8 NOT NULL DEFAULT,
      `Team` varchar (3) CHARACTER SET utf8 NOT NULL DEFAULT ',
      `UID` varchar (5) NOT NULL DEFAULT,
      `username` varchar (16) CHARACTER SET utf8 NOT NULL DEFAULT
```

Match_2016:

```
   `Week` varchar(4) CHARACTER SET utf8 DEFAULT NULL,
   `Date` varchar(22) CHARACTER SET utf8 DEFAULT NULL,
   `Team_1` varchar(37) CHARACTER SET utf8 DEFAULT NULL,
   `Team_1_id` varchar(5) CHARACTER SET utf8 DEFAULT NULL,
   `Team_2` varchar(34) CHARACTER SET utf8 DEFAULT NULL,
   `Team_2_id` varchar(5) CHARACTER SET utf8 DEFAULT NULL,
   `Score_1` int(11) DEFAULT NULL,
   `Score_2` int(11) DEFAULT NULL,
   `P_L` varchar(16) CHARACTER SET utf8 DEFAULT NULL,
   `Ru_L` varchar(16) CHARACTER SET utf8 DEFAULT NULL,
   `Re_L` varchar(19) CHARACTER SET utf8 DEFAULT NULL


   CREATE DEFINER=`cs4750s17yk7da`@`%` PROCEDURE `getTopREL`()
```

```
    READS SQL DATA
BEGIN
SELECT *
FROM (
      (SELECT
              (SELECT SUBSTRING_INDEX(`Re_L`, '.', 1)) AS `fname`,
              (SELECT SUBSTRING_INDEX(`Re_L`, '.', -1)) AS `lname`,
              COUNT(`Re_L`) AS `occurrence`
              FROM     `match_2016`
              WHERE SUBSTRING_INDEX(`Re_L`, '.', -1) != ''
              GROUP BY `Re_L`
              ORDER BY `occurrence` DESC) AS `freq_list`
JOIN (`Players`)
ON (
      (`Players`.`Name` LIKE CONCAT('%',`lname`,'%')) AND
      (`Players`.`Name` LIKE CONCAT(`fname`,'%'))
)
);
END$$ (Same for `getTopPL`(), `getTopRUL()`)
```

4. 3NF Proof

1) Players:

Calculating F+:

Functional Dependencies:

No → No

Name → Name

Pos → Pos

Ht → Ht

Wt → Wt

Class → Class

Hometown → Hometown

State → State

Team → Team

UID → UID

All the dependencies are trivial since all attributes are linearly independent of each other. It can be argued that Hometown and State can be considered linearly dependent, but for convenience factor, we considered the two attributes linearly independent.

Calculating Fc:

All dependencies are trivial, so Fc is the empty set.

Deriving 3NF tables from Fc:

Since all dependencies are trivial, the table containing all the attribute is 3NF-compliant.

2) School table (uva, vt, lib, nor, hamp, odu):

Calculating F+:

Functional Dependencies:

No → No

Name → Name

Pos → Pos

Ht → Ht

Wt → Wt

Class → Class

Hometown → Hometown

State → State

Team → Team

UID → UID

All the dependencies are trivial since all attributes are linearly independent of each other. It can be argued that Hometown and State can be considered linearly dependent, but for convenience factor, we considered the two attributes linearly independent.

Calculating Fc:

All dependencies are trivial, so Fc is the empty set.

Deriving 3NF tables from Fc:

Since all dependencies are trivial, the table containing all the attribute is 3NF-compliant.

3) Positional table (C, CB, DB, DE, DL, DT, FB, G, LB, LS, OL, OT, P, PK, QB, RB, S, TE, WR):

Calculating F+:

Functional Dependencies:

No → No

Name → Name

Pos → Pos

Ht → Ht

Wt → Wt

Class → Class

Hometown → Hometown

State → State

Team → Team

UID → UID

All the dependencies are trivial since all attributes are linearly independent of each other. It can be argued that Hometown and State can be considered linearly dependent, but for convenience factor, we considered the two attributes linearly independent.

Calculating Fc:

All dependencies are trivial, so Fc is the empty set.

Deriving 3NF tables from Fc:

Since all dependencies are trivial, the table containing all the attribute is 3NF-compliant.

4) Users:

Calculating F+:

Functional Dependencies:

username → username

password → password

admin → admin

All the dependencies are trivial since all attributes are linearly independent of each other.

Calculating Fc:

All dependencies are trivial, so Fc is the empty set.

Deriving 3NF tables from Fc:

Since all dependencies are trivial, the table containing all the attribute is 3NF-compliant.

5) Fav_players

Calculating F+:

Functional Dependencies:

Name → Name

Team → Team

UID → UID

username → username

All the dependencies are trivial since all attributes are linearly independent of each other.

Calculating Fc:

All dependencies are trivial, so Fc is the empty set.

Deriving 3NF tables from Fc:

Since all dependencies are trivial, the table containing all the attribute is 3NF-compliant.

6) match_2016

Calculating F+:

Functional Dependencies:

Week → Week

Date → Date

Team_1 → Team_1

Team_1_id → Team_1_id

Team_2 → Team_2

Team_2_id → Team_2_id

Score_1 → Score_1

Score_2 → Score_2

P_L → P_L

Ru_L → Ru_L

Re_L → Re_L

All the dependencies are trivial since all attributes are linearly independent of each other. It can be argued that Team_1 and Team_1_id along with Team_2 and Team_2_id can be considered linearly dependent, but for convenience factor, we considered these attributes linearly independent.

Calculating Fc:

All dependencies are trivial, so Fc is the empty set.

Deriving 3NF tables from Fc:

Since all dependencies are trivial, the table containing all the attribute is 3NF-compliant.

3. Evaluation of Product

1. Testing Procedure.

Sample queries listed in the section below were tested through the front end application and the database in phpMyAdmin. Checks, Triggers, and Stored Procedures were also tested through inserting rows and testing the behavior of the checks and triggers. The tests done are listed as the sample data and queries below.

2. Sample data and Sample Queries.

A. Logging in with the provided username.

COUNT(SELECT * FROM users WHERE username = $myusername AND password = $mypassword) → 1 for any successful login and 0 for unsuccessful login.

B. Insert a player into database

INSERT INTO `Players` (`No`, `Name`, `Pos`, `Ht`, `Wt`, `Class`, `Hometown`, `State`, `Team`, `UID`) VALUES

('1', 'DeCarlo Hamiltona', 'DT', ' 06-03', 335, 'FR', 'Plantation', 'FL', 'Lib', '3001'); →

1	DeCarlo Hamiltona	DT	06-03	335	FR	Plantation	FL	Lib	3001

Inserted into table Players, DT, and lib.

C. Deleting a player from database.

DELETE FROM `Players` WHERE Team = Lib AND UID = 3001→

1	DeCarlo Hamiltona	DT	06-03	335	FR	Plantation	FL	Lib	3001

Deleted from table Players, DT, and lib.

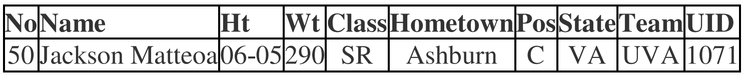
D. Searching a player

SELECT * FROM Players WHERE Name LIKE Hamiltona →

1	DeCarlo Hamiltona	DT	06-03	335	FR	Plantation	FL	Lib	3001

E. Searching a position

SELECT * from C WHERE Name LIKE Matteoa→



No	Name	Ht	Wt	Class	Hometown	Pos	State	Team	UID
50	Jackson Matteoa	06-05	290	SR	Ashburn	C	VA	UVA	1071

F. Searching by school

SELECT * FROM UVA WHERE Name LIKE Abdullaha→

No	Name	Ht	Wt	Class	Hometown	Pos	State	Team	UID
--	Naji Abdullaha	6-5	235	FR	Jacksonville	DE	FL	UVA	1001

G. Search Players who has been Receiving Leader in at least one of the matches

(using stored procedure)

CALL getTopPL()→

fname	lname	occ	No	Name	Pos	Ht	Wt	Class	Hometown	State	Team	UID
D	Brown	5	44	Dia'Vante Browna	DE	06-02	245	JR	Greensboro	NC	Lib	3051
J	Smith	1	83	Justin Smitha	WR	06-02	165	FR	Richmond	VA	nor	4072
D	Davis	10	--	Dillon Davis	LB	06-02	210	FR	Bellaire	TX	UVA	1010
K	Benkert	5	6	Kurt Benkert	QB	06-04	230	JR	Cape Coral	FL	UVA	1032
M	Johns	1	15	Matt Johns	QB	06-05	215	SR	Chalfont	PA	UVA	1040
J	Evans	8	4	Jerod Evans	QB	06-04	235	JR	Dallas	TX	VT	2005
T	Jackson	5	56	T.J. Jackson	OL	06-03	330	FR	Cumberland	VA	VT	2072
T	Hill	1	94	Trevon Hill	DE	06-05	234	FR	Virginia Beach	VA	VT	2112