CS 353

DATABASE SYSTEMS

Spring 2018

ÇiftKale

A Football Database System

FINAL REPORT DOCUMENT

Prepared By:

Hakan Türkmenoğlu

Orkun Alpar

Selim Fırat Yılmaz

Alper Şahıstan

1. Description of ÇiftKale:

ÇiftKale is a football management database system which is used by football fans, club directors and player agents alike. It includes information about players, agents, clubs, leagues and coaches. It also includes news about transfers, match results along with statistics(such as assists, shoots, goals, passing, fouls and cards). It also has league standings.It allows its users to create profiles and track statistics about their teams and players of interest. It provides an extensive player transfer/offer system. Offer system uses buckets and that contains the offers made by Club directors. Club directors can offer money or players in exchange of other club's player or money. A club cannot exceed its annual budget while making offers. Sportsman is the generalization of Player and Coach. Sportsman profiles include a picture of the sportsman, name, age, nationality, current team, annual wage and if possible, social media accounts. Main distinction between coaches and players is different statistics.

2. Separation of Labour:

2.1. Work Done by Selim Fırat Yılmaz:

- Lead front-end developer.
- SMS notification system integration.
- Fixes on some existing views.
- Development of some SQL queries.
- Writing of some parts of design report.
- Back-end and front-end integration.

2.2. Work Done by Hakan Türkmenoğlu:

- Developed of most of the SQL queries
- Co-designed and implemented the bucket-based offer system.
- Back-end and front-end integration.
- Lead designer of the database and relational schemas.
- Revised the E/R from proposal report.
- Fixed some back-end bugs.

2.3. Work Done by Orkun Alpar:

- Developed a Python script to insert data sets to our database.
- Wrote of some parts of the final report.
- Developed of some SQL queries.
- Co-designed of the bucket-based offer system.
- Tested of some back-end services.

2.4. Work Done by Alper Şahıstan:

- Methods for raw SQL calls.
- Pagination and filtering at back-end.
- Fixed some old SQL queries and wrote additional SQL queries.
- Revised the E/R from proposal report.
- Wrote some parts of the final report.

3. Final E/R:

Note: In our feedback I was advised that we should include a user type as an attribute for person entity however we later realised that it was unnecessary since each user had a separate table in our SQL implementation.

4. Final List of Table Schemas:

Primary keys are underlined in the given relational models:

- Person

  Person(username, photo, first_name, last_name, hashed_password, e_mail,phone_number, date_of_registration)

- Agent

  Agent(agent_username)

  FOREIGN KEY (agent_username) REFERENCES Person(username)

- Sportsman

  Sportsman(sportsman_username, date_of_birth, salary)

  FOREIGN KEY (sportsman_username) REFERENCES Person(username)

- Player

  Player(player_username, position, kit_number, weight, height, dominant_foot,
  shot_accuracy, overall_score, nationality, agent_username)

  FOREIGN KEY (player_username) REFERENCES Sportsman(sportsman_username)

  FOREIGN KEY (agent_username) REFERENCES Agent(agent_username)

- Coach

  Coach(coach_username, experience)

  FOREIGN KEY (coach_username) REFERENCES

Sportsman(sportsman_username)

- Director

  Director(director_username)

  FOREIGN KEY (director_username) REFERENCES
  Sportsman(sportsman_username)

- Offer

  Offer(offer_id, date, price, status, director_sender, director_receiver)

  FOREIGN KEY (director_receiver) REFERENCES
  Director(director_username)

  FOREIGN KEY (director_sender) REFERENCES
  Director(director_username),

- Bucket

  Bucket(offer_id, player_username)

  FOREIGN KEY (player_username) REFERENCES
  Player(player_username)

- League

  League(league_name, league_start, league_end)

- LeagueCountry

  LeagueCountry(league_name, league_country)

- LeagueSponsor

  LeagueSponsor(sponsor_name, league_name, league_start)

  FOREIGN KEY (league_name, league_start) REFERENCES
  League

- Club

  Club(club_name, short_name, logo_photo, since, budget, value, stadium, country, league_name, league_start, standing)

  FOREIGN KEY (league_name, league_start) REFERENCES League

- ClubSponsor

  ClubSponsor(sponsor_name, club_name)

  FOREIGN KEY (club_name) REFERENCES Club

- WorksFor

  WorksFor(sportsman_username, club_name, start_date, end_date)

  FOREIGN KEY (sportsman_username) REFERENCES Sportsman

  FOREIGN KEY (club_name) REFERENCES Club

- Match

  Match(datetime, stadium, overtime, referee, home, away, home_goals(), away_goals(), league _name, league_start)

  FOREIGN KEY (home) REFERENCES Club(club_name)

  FOREIGN KEY (away) REFERENCES Club(club_name)

  FOREIGN KEY (league_name, league_start) REFERENCES League

- Stat

Stat(player_username, time, description, match_time, stadium)

FOREIGN KEY (player_username) REFERENCES Player(player_username)

FOREIGN KEY (match_time, stadium) REFERENCES Match(datetime, stadium)

5. Implementation Details:

Our architecture consists of three parts: database, backend and frontend. This modularity helped us to share the work.

In database, simply we store our data and this data is only accessible by the backend.

In backend, we handled the communication between frontend and database and handles complex operations such as forgot password securely. We chose to use Django Framework backend in Python to create our REST API.

In frontend, we used CoreUI and React Framework better user experience.

React Framework frontend. We also chose PostgreSQL as our database due to its compatibility with SQL standards. In backend, we used REST API structure to communicate with front-end and to separate front-end from the backend. Simply, frontend requests an information via GET or posts a form/input data via POST request in HTTP protocol. This request is met by our backend and gives the results in JSON format. This format is easily converted to backend. name_tools library was used to distinguish between titles first/middle names and surnames of Sportsmen such as director, player, coach. Our SMS information system using AWS SNS deployed on AWS Lambda. We also used react-table to handle retrieving list of information at server-side. (Details are explained in Table Search/Pagination/Sorting in Design Report)

As for our database, we needed to import actual data, in order to test the functionalities of our system. Therefore, we used a player database which consisted of more than 15k players, along with their personal info, team info and FIFA stats. However, as expected, the dataset didn't fully match with our attributes. In order to import, we have used a Python script that gets player information row-by-row from our dataset and created instances of players, teams, leagues, et cetera. We also needed to get league country from the league name (e.g. Spanish Primera División -> Spain). In order to tackle that, we again used a table that maps nationalities with nations.

Technical Problems During Development

Automatic Deployment

Since we realized that it takes much time, to speed up our development, we deployed our backend to Heroku and set an auto deployment webhook to be deployed and built when a commit to Github repository occurs. Also, we set live-reload for frontend so that when a file changes, it refreshes the browser immediately.

Cross Origin Error

We encountered famous "Cross origin not allowed" while trying to access our REST API from our locals. We solved it after long research via adding Cross-origin-allow: * header to our backend.

Lack of Football Datasets

We wanted to import football dataset to our database but there are not many database containing the information we want due to commercial issues. Thus, we sometimes inserted dummy data to test our transfer market system where we couldn't find its alternative.

Python Version Issues

We have decided that we'll be using Python for our backend development. After this decision was made, we have divided some work among us but later on we realized that we had some integration problems. It turned out to be caused by different Python versions utilized by our team members.

6. Advanced Database features:

Views:

A lot of the times we need to access a Sportsman's current club name. This is addressed by the CurrentOccupations view:

```
CREATE VIEW CurrentOccupations AS (SELECT * FROM WorksFor
WHERE end_date IS NULL
);
```

We have also declared PlayerDirector view that matches a player's current director.

```
CREATE VIEW PlayerDirector AS (
SELECT p.player_username, d.director_username
FROM player p, director d, CurrentOccupations cop, CurrentOccupations cod WHERE
p.player_username = cop.sportsman_username AND
d.director_username = cod.sportsman_username AND
cop.club_name = cod.club_name );
```

Triggers:

The following trigger updates budgets of the clubs whenever an offer is accepted.

```
CREATE OR REPLACE FUNCTION OfferTrigger()
  RETURNS trigger AS
$$
BEGIN
   UPDATE Club SET budget = budget + NEW.price
        WHERE club_name = (SELECT wf.club_name FROM WorksFor wf, Director d
                                        WHERE wf.sportsman_username =
d.director_username
                                        AND wf.end_date IS NULL
                                        AND d.director_username = NEW.director_receiver);
        UPDATE Club SET budget = budget - NEW.price
        WHERE club_name = (SELECT wf.club_name FROM WorksFor wf, Director d
                                        WHERE wf.sportsman_username =
d.director_username
                                        AND wf.end_date IS NULL
                                        AND d.director_username = NEW.director_sender);
   RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';

CREATE TRIGGER offertrigger
BEFORE INSERT OR UPDATE ON Offer
FOR EACH ROW
WHEN (NEW.status = 'accepted')
EXECUTE PROCEDURE OfferTrigger();
```

Reports:

The following stored procedure is executed for each league to generate reports. The procedure is dynamically generated and executed. The procedure calculates the number of offers and their sum of prices for a given league and year.

```
CREATE OR REPLACE FUNCTION ReportLeagueStats(league_name VARCHAR(32), year
INT)
        RETURNS TABLE(cnt BIGINT, sumprice BIGINT) as $$
        BEGIN
                RETURN QUERY EXECUTE
                format ('SELECT COUNT(*), SUM(ABS(o.price)) FROM Offer o, Club c1,
Club c2, CurrentOccupations co1, CurrentOccupations co2
                WHERE status = "accepted"
                AND date_part("year", date) = %s
                AND co1.club_name = c1.club_name
                AND co2.club_name = c2.club_name
```

```
            AND co1.sportsman_username = o.director_sender
            AND co2.sportsman_username = o.director_receiver
            AND (c1.league_name = ''%s'' OR c2.league_name = ''%s'')', year,
league_name, league_name);
    END;
    $$ LANGUAGE plpgsql;
```

Example:
SELECT * FROM ReportLeagueStats('Turkish Süper Lig', 2018);
gives
11, 143305

Indexes:
To improve the performance of the database, we have implemented a few secondary indexes:

```
CREATE INDEX phone_number_index ON Person(phone_number);
CREATE INDEX email_index ON Person(email);
CREATE INDEX offer_status_index ON Offer(status);
```

7. User Manual:

7.1. Register Page:

Register page allows users to register for an account. Later by demand and verification they can be **upgraded to director, agent, coach or player** . It takes firstname, lastname email, phone number and password. After providing these information, user should click “create account” “button, in order to proceed. Using this view, one can also go to “login” and “forgot password” screens.

7.2. Login Page:

Users can login by entering their username and password and pressing login button. Each user can manage their accounts after this action.

7.3. Forgot Password Page:

Allows users who forgot their passwords to reset their passwords by entering their registered phone numbers. After entering the registered phone number, user should click “reset password” button.
After a short while if the number is correct user receives a SMS text with their new password .

7.4. Change Password Page:

This page allows logged in users to change their passwords.By entering their old passwords, new one and pressing change password button.

7.5. Change Username Page:

This page allows users to change their username yet username should be unique since it’s our primary key. After entering the new username and password they can press Change Username button to change their usernames.

7.6. Change Profile Picture Page:

User can swap a profile picture by pressing the change file button and accessing their file system. After selecting the new profile picture Change Your Photo button can be pressed to apply the changes.

7.7. Delete Own Account Page:

Users can delete their account by typing their password and pressing the button down bellow.

7.8. Your Pending Offers(To accept) Page

Directors can view their offers that are made to their clubs. And can decline or accept the offers made to their clubs by clicking on the “accept” and “decline” buttons, respectively.

7.9. Make Offer To a Team

Directors can make offers to the teams or players of others teams. This can be done by special buttons that are visible only to director type accounts. When directors make offer for another team they choose their offered player(s) and opposing team's desired players. And click to button down below to proceed with the offer. When they make offer with money they select the desired players and enter the amount of money they offer for the player(s) and click button down below to proceed. Money can be negative that indicates that they offer a player from their own team and ask for money in return.

7.10. Leagues/Clubs/Players List View

List views are used for each of Clubs, Leagues and Players. The user can sort ascendingly and descendingly each of the attributes. If needed, multiple attributes can be sorted at the same time using SHIFT key. Multiple attributes can be filtered case insensitively using the textboxes under the column headers.By clicking on a desired coach/player/team/league users can go that coach/player/team/league's page

7.11. General Navigation

Each section is divided into three categories called "Authentication", "Account" and "Football". Account related information is not shown when the user is not logged in. Clicking the sections will reload the user interface without refreshing the page thus providing a single page application to the user that is akin to desktop application experience.

8. Website

Our project information website containing reports and project description is below:
https://selimfirat.github.io/ciftkale/
http://github.com/selimfirat/ciftkale