Soccer League DBMS

CPS 510
Dr. Abdolreza Abhari

November 30, 2018

Prabagar Sivakumar 500765715
Sagar Singh Punn 500761087
George Nakhla 500756066

TABLE OF CONTENTS

Introduction

The DBMS we created is for a soccer league. It's main priority is to manage and track various different statistics of teams and players in the league. In order to accomplish this, we have created 4 different entities that we will be using throughout the DBMS. We have not finalized the different relationships we will be having since we are currently removing repetitive or unnecessary relationships we have, to optimize our Entity Relationship Diagram. For users we have currently Fans, Team president, Match officials. They will all have a different external use, which means some entities' attributes will be restricted to certain uses.

There will be relationships between, certain entities and the attributes within them, for example the entity Team will be able to have attributes of player, like player name, and team ID, and goals scored.

Functions:

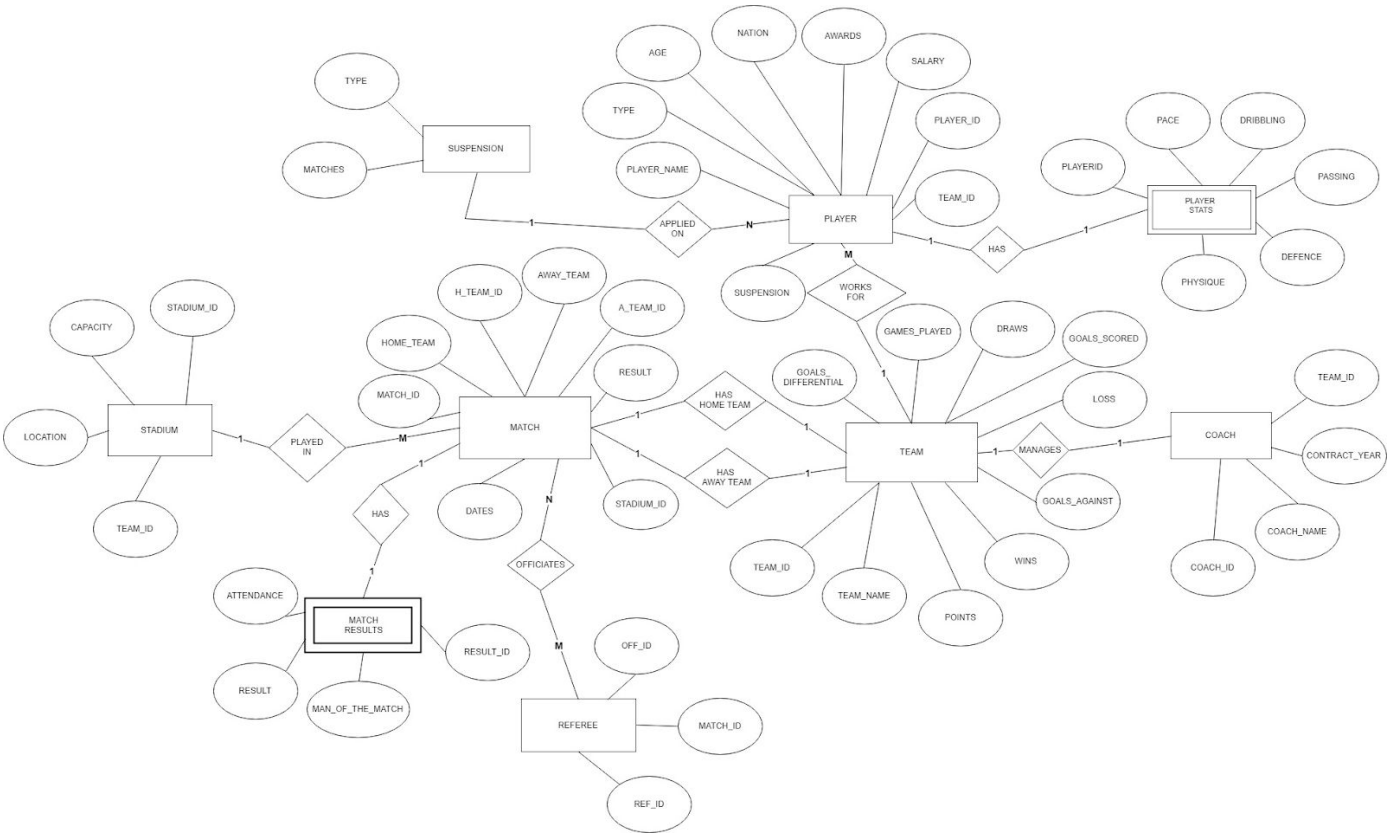
1. Search Function - Ability to search the statistics of a specific team or player.
    a. Search individual players.
2. Table Modifications - Ability to Remove or Update a table's row
3. Player Summary - Ability to see information.

Information Expected:
In order to summarize the information that we expect, we have given a sample of the 4 different entities involved in our DBMS along with their corresponding attributes.

Data types and its description

1. Integer: one optional sign character (+ or -) followed by at least one digit(0-9). Leading and trailing blanks are ignored. No other character is allowed.
2. Varchar: it is used to store alphanumeric characters. In this data type we can set the maximum number of characters up to 8000 ranges by defaults SQL server will set the size to 50 characters range.

ER Model


TYPE
MATCHES
SUSPENSION
AGE
NATION
AWARDS
SALARY
TYPE
PLAYER_ID
PLAYER_NAME
TEAM_ID
PLAYER
APPLIED ON
1
N
M
HAS
PLAYERID
PACE
DRIBBLING
PASSING
PLAYER STATS
DEFENCE
PHYSIQUE
SUSPENSION
WORKS FOR
GAMES_PLAYED
DRAWS
GOALS_SCORED
LOSS
GOALS_ DIFFERENTIAL
TEAM
MANAGES
COACH
TEAM_ID
CONTRACT_YEAR
COACH_NAME
COACH_ID
GOALS_AGAINST
WINS
POINTS
TEAM_NAME
TEAM_ID
CAPACITY
STADIUM_ID
LOCATION
STADIUM
TEAM_ID
PLAYED IN
H_TEAM_ID
AWAY_TEAM
A_TEAM_ID
HOME_TEAM
RESULT
MATCH_ID
MATCH
HAS HOME TEAM
HAS AWAY TEAM
STADIUM_ID
DATES
HAS
OFFICIATES
ATTENDANCE
MATCH RESULTS
RESULT_ID
RESULT
MAN_OF_THE_MATCH
OFF_ID
REFEREE
MATCH_ID
REF_ID

Schema design

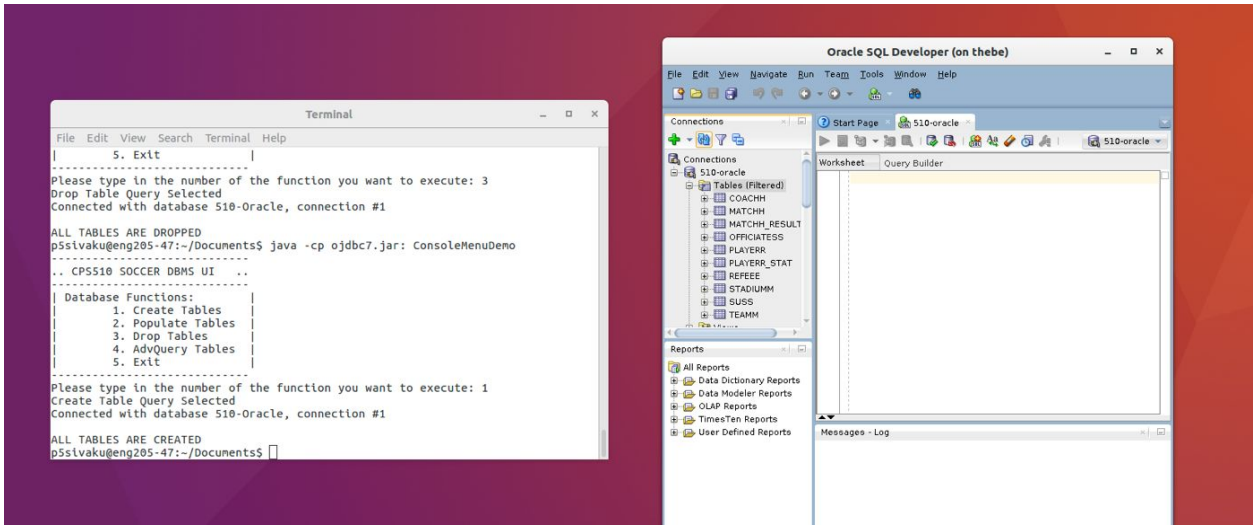
```
CREATE TABLE "SPUNN"."SUSPENSION"
  (  "PLAYER_ID" NUMBER,
  "DURATION" NUMBER ,
PRIMARY KEY (PLAYER_ID),
FOREIGN KEY(PLAYER_NAME,PLAYER_ID)
  );
 CREATE TABLE "SPUNN"."COACH"
  (  "COACH_NAME" VARCHAR2(20 BYTE),
“COACH_ID” NUMBER,
  "TYPE" VARCHAR2(20 BYTE),
  "TEAM_ID" NUMBER(25,0) ,
  "CONTRACT" VARCHAR2(20 BYTE),
  "SALARY" NUMBER(25,0) ,
PRIMARY KEY (COACH_NAME, COACH_ID),
FOREIGN KEY(TEAM_NAME,TEAM_ID)
  );
 CREATE TABLE "SPUNN"."LEAGUE"
  (  "TEAMS" VARCHAR2(25 BYTE),
  "MATCHS" VARCHAR2(25 BYTE),
PRIMARY KEY (TEAMS)
  );
 CREATE TABLE "SPUNN"."MATCH_RESULTS"
  (  "MATCH_ID" NUMBER(25,0) ,
  "ATTENDANCE" NUMBER(25,0) ,
  "RESULT" VARCHAR2(20 BYTE),
  "MOTM" VARCHAR2(20 BYTE),
  "POSSESSION" VARCHAR2(20 BYTE),
FOREIGN KEY (MATCH_ID)
```

```
 ) ;
CREATE TABLE "SPUNN"."MATCH"
  (  "MATCH_ID" NUMBER(25,0) ,
  "DATES" NUMBER(25,0) ,
  "STADIUM_ID" NUMBER(25,0) ,
  "HOME_TEAM" VARCHAR2(20 BYTE),
  "AWAY_TEAM" VARCHAR2(20 BYTE),
PRIMARY KEY (MATCH_ID),
FOREIGN KEY (STADIUM_ID)
  );
CREATE TABLE "SPUNN"."PLAYER_STAT"
  (  "PLAYER_NAME" VARCHAR2(25 BYTE),
  "PLAYERID" NUMBER,
  "PACE" NUMBER,
  "DRIBBLING" NUMBER,
  "COLUMN1" NUMBER,
  "DEFENSE" NUMBER,
  "PHYSIQUE" NUMBER,
PRIMARY KEY (PLAYER_NAME),
FOREIGN KEY (PLAYER_NAME,PLAYER_ID)
  );
CREATE TABLE "SPUNN"."PLAYER"
  (  "PLAYER_NAME" VARCHAR2(25 BYTE),
“PLAYER_ID” NUMBER,
  "TYPE" VARCHAR2(25 BYTE),
  "AGE" NUMBER,
  "NATION" VARCHAR2(25 BYTE),
  "TEAM_NAME" VARCHAR2(25 BYTE),
  "AWARDS" NUMBER,
```

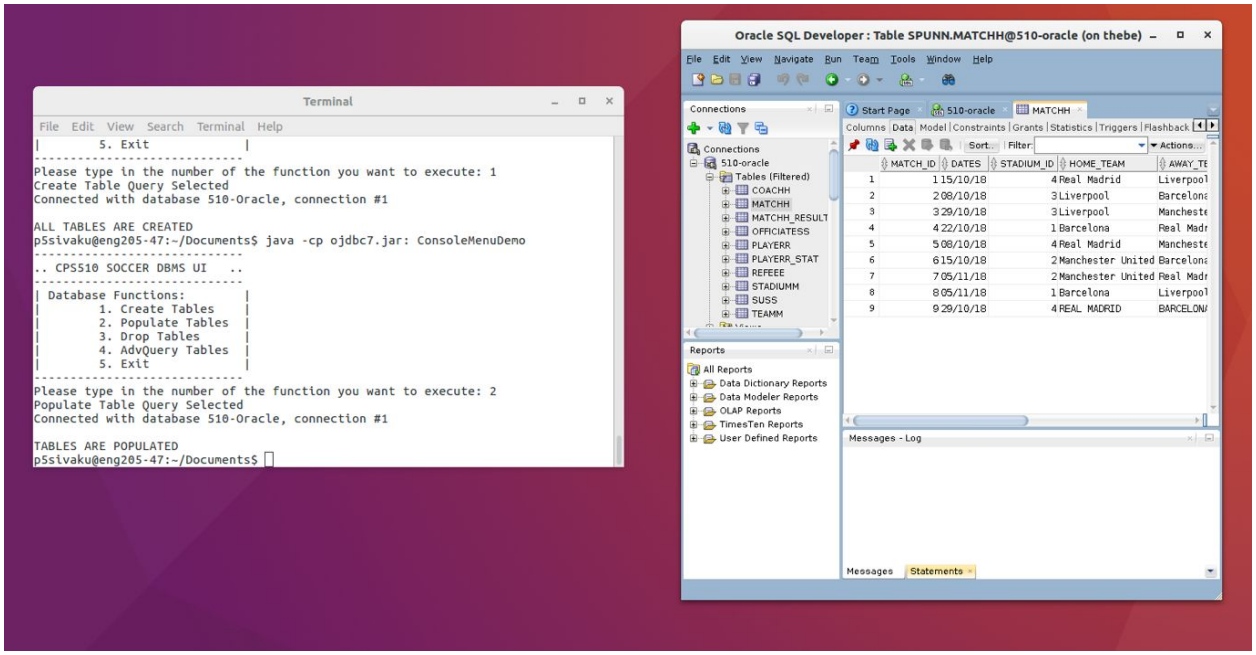
```
  "SALARY" NUMBER,
  "PLAYER_ID" VARCHAR2(20 BYTE),
PRIMARY KEY (PLAYER_NAME,PLAYER_ID),
FOREIGN KEY (TEAM_NAME,TEAM_ID)
  );
CREATE TABLE "SPUNN"."REFEREE"
  (  "REF_ID" NUMBER,
  "TYPE" VARCHAR2(25 BYTE),
PRIMARY KEY (REF_ID)
  );
CREATE TABLE "SPUNN"."STADIUM"
  (  "STADIUM_ID" NUMBER,
  "HOME_TEAM" VARCHAR2(25 BYTE),
  "CAPACITY" NUMBER ,
  "LOCATION" VARCHAR2(25 BYTE),
PRIMARY KEY (STADIUM_ID)
  );
CREATE TABLE "SPUNN"."TEAM"
  (  "TEAM_NAME" VARCHAR2(20 BYTE) ,
  "GAMES_PLAYED" NUMBER ,
“TEAM_ID” NUMBER,
  "WINS" NUMBER ,
  "LOSS" NUMBER ,
  "DRAWS" NUMBER ,
  "GOALS_SCORED" NUMBER ,
  "GOAL_DIFFERENTIAL" NUMBER ,
  "POINTS" NUMBER ,
  "GOALS_AGAINST" NUMBER ,
PRIMARY KEY (TEAM_NAME,TEAM_ID) );
```

Unix shell advanced query demo

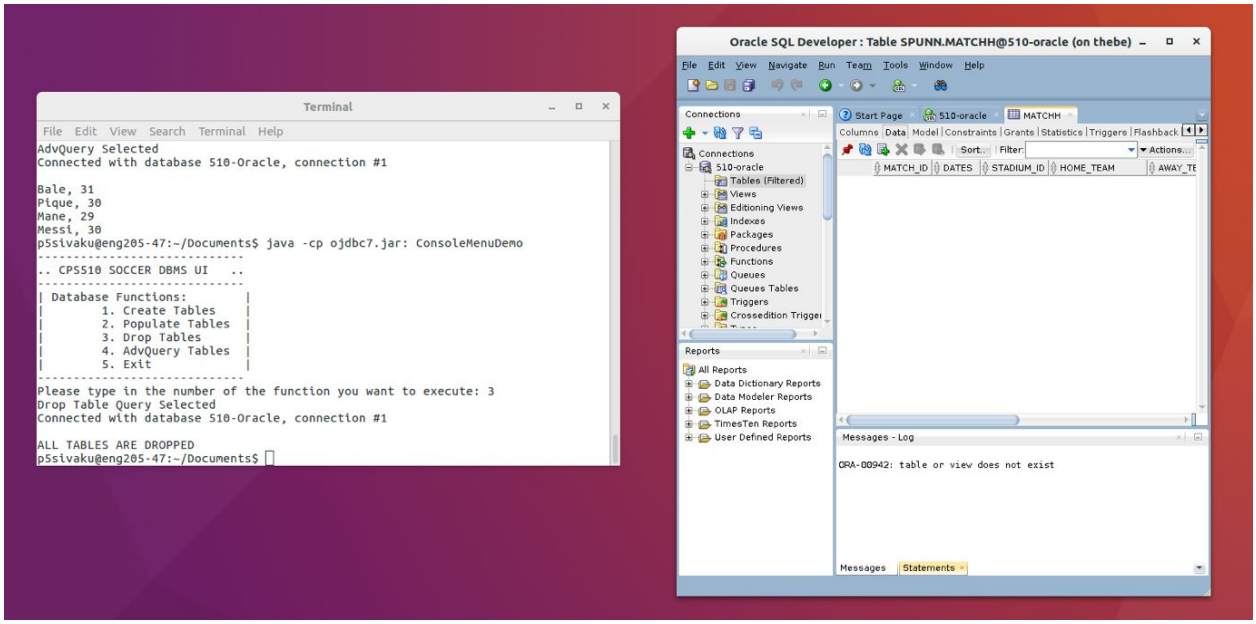
1. Creating all the tables.



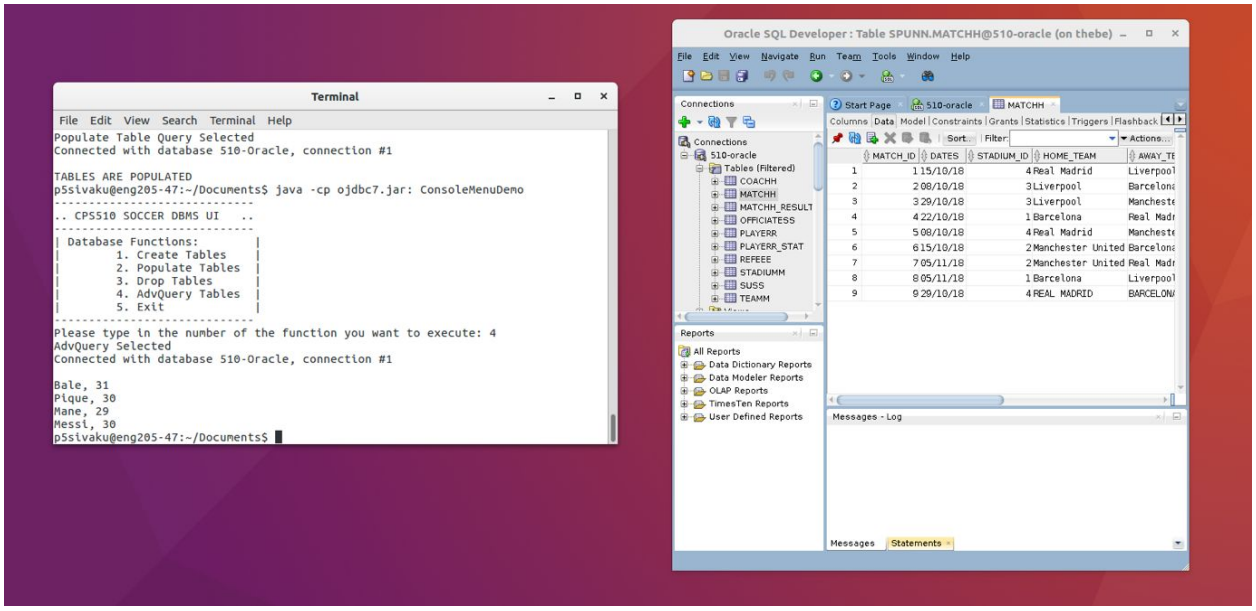
2. Populating all the tables.



3. Dropping all the tables.



4. Advance queries.



Simple Queries

SELECT * FROM MATCHH
SELECT DISTINCT MATCH_ID, DATES, STADIUM_ID, HOME_TEAM, AWAY_TEAM FROM MATCH;
INSERT INTO MATCHH (MATCH_ID, DATES, STADIUM_ID, HOME_TEAM, AWAY_TEAM) VALUES (9, '29/10/18', 4, 'Real Madrid', 'Barcelona')

SELECT * FROM MATCHH_RESULTS
SELECT DISTINCT MATCH_ID, ATTENDANCE, RESULT, FROM MATCHH;
INSERT INTO MATCHH_RESULTS (MATCH_ID, ATTENDANCE, RESULT) VALUES (8, 96389, '2-2')

SELECT * FROM PLAYER
SELECT DISTINCT NAME , TYPE , AGE, NATION ,TEAM_NAME, AWARDS, SALARY, PLAYER_ID, TEAM_ID FROM MATCHH;
INSERT INTO MATCH (NAME, TYPE ,AGE, NATION, TEAM_NAME, AWARDS, SALARY, PLAYER_ID, TEAM_ID) VALUES ('De Gea', 'Goalkeeper', '26', Manchester United', 74994, 'Old Trafford')

SELECT * FROM PLAYER_STAT
SELECT DISTINCT PLAYERNAME, PLAYERID, PACE, DRIBBLING, PASSING, DEFENCE, PHYSIQUE FROM PLAYER_STAT;
INSERT INTO PLAYER_STAT (PLAYERNAME, PLAYERID, PACE, DRIBBLING, PASSING, DEFENCE, PHYSIQUE) VALUES ('De Gea', 4, 88, 90, 95, 79, 70)

SELECT * FROM STADIUM
SELECT DISTINCT STADIUM_ID, CAPACITY, LOCATION, TEAM_ID FROM STADIUM;
INSERT INTO MATCH_RESULTS (STADIUM_ID, CAPACITY, LOCATION, TEAM_ID) VALUES (2, 'Manchester United', 74994, 'Old Trafford')

SELECT * FROM TEAM
SELECT DISTINCT TEAM_NAME, GAMES_PLAYED, TEAM_ID, WINS, LOSS, DRAWS, GOALS_SCORED, GOAL_DIFFERENTIAL, POINTS GOALS_AGAINST FROM MATCHH;
INSERT INTO MATCHH (TEAM_NAME, GAMES_PLAYED, TEAM_ID, WINS, LOSS, DRAWS, GOALS_SCORED, GOAL_DIFFERENTIAL, POINTS GOALS_AGAINST) VALUES ('Manchester United', 4,4,1,2,1,3,-4,2,7)

Advanced Queries

```
//finds a specific coach, player and team all from the same ID of 2
SELECT c.COACH_NAME,PLAYER_NAME,t.TEAM_NAME
 FROM  TEAM t, PLAYER p, COACH c
 WHERE t.TEAM_ID = 2
       AND p.TEAM_ID = 2
       AND c.TEAM_ID = 2
ORDER BY PLAYER_ID ASC;

// finds all players between the ages of 29 and 31
SELECT * FROM "PLAYER"
WHERE AGE >=29
     AND AGE <=31;

  //finds the average age of all players contained in the entity PLAYER
SELECT 'Average age of players is', AVG(AGE)
FROM "PLAYER";
```

Functional dependencies:

WORK FOR [M:1]
Work_Id —> {player_id, team_id}
Player_Id —> {work_id, team_id}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF but there is transitive dependency.

Updated FD:
Work_Id —> {player_id, team_id}

TEAM [Entity]
Team_id —> {team_name, games_played, wins, loss, draws, goal_scored, goal_differential, points, goal_against}

1NF:Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF and there is no transitive dependency.

STADIUM[Entity]
Stadium_id —> {team_id, capacity, location}
team_id —> {stadium_id, capacity, location}

1NF:Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF but there is transitive dependency.

Updated FD:
Stadium_id —> {team_id, capacity, location}

REFEREE[Entity]
ref_id —> {ref_type, ref_name}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).

3NF: It is 2NF and there is no transitive dependency.

PLAYER_STATS[Entity]
PlayerID —> {pace, dribbling, passing, defense, physique}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF and there is no transitive dependency.

PLAYER[Entity]
Player_id —> { Player_name, type, age, nation, team_name, awards, salary, team_id}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF and there is no transitive dependency.

PLAYED_IN[1:1]
played_in_id —> {match_id, stadium_id}
Match_id —> {played_in_id, stadium_id}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF but there is transitive dependency.

Updated FD:
played_in_id —> {match_id, stadium_id}

OFFICIATES [M:1]
off_id —> {match_id, ref_id}
{match_id, ref_id} —> off_id

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF but there is no transitive dependency.

MATCH_RESULTS[Entity]
match_id —> {attendance, results, motm}

1NF: Each column in this table has atomic values.

2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF and there is no transitive dependency.

MATCH[Entity]
match_id —> {date, stadium_id, home_team, away_team, h_team_id, a_team_id}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF and there is no transitive dependency.

MANAGES[1:1]
manages_id —> {coach_id, team_id}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF and there is no transitive dependency.

COACH[Entity]
Coach_id —> {coach_name, contract_year, team_id}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).

3NF: It is 2NF and there is no transitive dependency.

Normalization into 3NF

WORK FOR [M:1]
Work_Id —> {player_id, team_id}
Player_Id —> {work_id, team_id}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF but there is transitive dependency.

Updated FD:
Work_Id —> {player_id, team_id}

TEAM [Entity]
Team_id —> {team_name, games_played, wins, loss, draws, goal_scored, goal_differential, points, goal_against}

1NF:Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF and there is no transitive dependency.

STADIUM[Entity]
Stadium_id —> {team_id, capacity, location}
team_id —> {stadium_id, capacity, location}

1NF:Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF but there is transitive dependency.

Updated FD:
Stadium_id —> {team_id, capacity, location}

REFEREE[Entity]
ref_id —> {ref_type, ref_name}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF and there is no transitive dependency.

PLAYER_STATS[Entity]
PlayerID —> {pace, dribbling, passing, defense, physique}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF and there is no transitive dependency.

PLAYER[Entity]
Player_id —> { Player_name, type, age, nation, team_name, awards, salary, team_id}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF and there is no transitive dependency.

PLAYED_IN[1:1]
played_in_id —> {match_id, stadium_id}
Match_id —> {played_in_id, stadium_id}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF but there is transitive dependency.

Updated FD:
played_in_id —> {match_id, stadium_id}

OFFICIATES [M:1]
off_id —> {match_id, ref_id}
{match_id, ref_id} —> off_id

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF but there is no transitive dependency.

MATCH_RESULTS[Entity]
match_id —> {attendance, results, motm}

1NF: Each column in this table has atomic values.

2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF and there is no transitive dependency.

MATCH[Entity]
match_id —> {date, stadium_id, home_team, away_team, h_team_id, a_team_id}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF and there is no transitive dependency.

MANAGES[1:1]
manages_id —> {coach_id, team_id}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF and there is no transitive dependency.

COACH[Entity]
Coach_id —> {coach_name, contract_year, team_id}

1NF: Each column in this table has atomic values.
2NF: It is 1NF and there is no composite key. It only has one primary key (one col).
3NF: It is 2NF and there is no transitive dependency.

Normalization into BCNF:

TEAM [Entity]
Team_id —> {team_name, games_played, wins, loss, draws, goal_scored, goal_differential, points, goal_against}

Is BCNF. For the FD in this table, all candidate keys are on the left side of the FDS. In other words,we have one primary key only pointing to each attribute.

STADIUM[Entity]
Stadium_id —> {team_id, capacity, location}

Is BCNF. For the FD in this table, all candidate keys are on the left side of the FDS. In other words,we have one primary key only pointing to each attribute.

REFEREE[Entity]
ref_id —> {ref_type, ref_name}

Is BCNF. For the FD in this table, all candidate keys are on the left side of the FDS. In other words,we have one primary key only pointing to each attribute.

PLAYER_STATS[Entity]
PlayerID —> {pace, dribbling, passing, defense, physique}

Is BCNF. For the FD in this table, all candidate keys are on the left side of the FDS. In other words,we have one primary key only pointing to each attribute.

MATCH_RESULTS[Entity]
match_id —> {attendance, results, motm}

Is BCNF. For the FD in this table, all candidate keys are on the left side of the FDS. In other words,we have one primary key only pointing to each attribute.

MATCH[Entity]
match_id —> {date, stadium_id, home_team, away_team, h_team_id, a_team_id}

Is BCNF. For the FD in this table, all candidate keys are on the left side of the FDS. In other words,we have one primary key only pointing to each attribute.

COACH[Entity]
Coach_id —> {coach_name, contract_year, team_id}

Is BCNF. For the FD in this table, all candidate keys are on the left side of the FDS. In other words,we have one primary key only pointing to each attribute.

Player[Entity]

Was not BCNF.

Relationship Algebra

1. Select all the players in the Database from players table.
    a. **PLAYERS**
2. Select all the players older than 30.
    a. $\sigma_{AGE > 30}$ **(PLAYERS)**
3. Select
    a. $\sigma_{MIN(PACE),MAX(PASSING)}$ **(PLAYERS)**
4. Select Coach's contact year.
    a. $\Pi_{contract\_year}$**(COACH)**
5. Select date and results for Real Madrid as Home Team.
    a. $\Pi_{DATES,HOME\_TEAM,RESULT}(\sigma_{HOME\_TEAM= 'REAL\_MADRID'}$ **(MATCH))**
6. Select the all the Results for Barcelona where attendance greater than 50000.
    a. **fromResult ←** $\Pi_{RESULT}(\sigma_{ATTENDANCE < 50000}$**(MATCH_RESULTS))**
    b. inMatch ← $\Pi_{RESULT}(\sigma_{HOME\_TEAM = 'BARCELONA}$**(MATCH))**
    c. Result ← inMatch - fromResult
7. Average Salary of the players
    a. $F_{AVERAGE\ SALARY}$ **(PLAYERS)**
8. List Players older than 25 from Barcelona or Players older than 30 from Real Madrid.
    a. $\Pi_{PLAYER\_ID,PLAYER\_NAME}(\sigma_{(AGE>25\ AND\ TEAM\_ID= 'BARCELONA') OR (AGE>30\ AND\ TEAM\_ID='REAL\_MADRID')}$**(MATCH)**
9. Select Referee id and their type who officiated Barcelona match.
    a. $\Pi_{REF\_ID, TYPE}(\sigma_{HOME\_TEAM = 'BARCELONA}$ **(Referee ⋈ Officiates ⋈ Match)**
10. Select Liverpool Match Result and Man of the Match.
    a. $\Pi_{RESULT, MAN\_OF\_THE\_MATCH}(\sigma_{HOME\_TEAM = 'LIVERPOOL'}$ **(Match ⋈ Match_Results)**

Conclusion

In this project for Soccer Database Management System, after following every step through the course of the labs, we achieved perfect state of DBMS. Completing the assignments gave us knowledge regarding redundancies in our system, which were removed. Majority of the correction to our tables came from Normalization projects where BCNF was the final normalization performed through the use of various algorithms such as Bernstein's Algorithm. Over all the project allowed us to reflect on our past assignments when we had to go back and fix the diagrams and models to match our updated database system. The figure below is the final state of our system in Java GUI.